Introduction to MHD Simulation

Akira Kageyama

Graduate School for System Informatics,
Kobe University,
Japan

2011.02.15 Invited Lecture at *Sokendai* Asian Winter School, NIFS

Goal

- To understand basic ideas of MHD simulation
- Introductory course
- Many simple/sample/example codes.
- I'll make the source codes available for you.
  - Check my lab's website:
    URL: http://www.research.kobe-u.ac.jp/csi-viz/
  - The lecture note (this pdf file) will be available, too at the above URL and this school's web page.

Outline

- Speed of computers
- How can we make use of computer's speed?
- Numerical methods
- Sample 1-D simulations

TOP500 List

TOP500 List - November 2010 (1-100)

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

Power data in KW for entire system

URL: http://www.top500.org/list/2010/11/100

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	National Supercomputing Center in Tianjin China	Tianhe-1A - NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C / 2010 NUDT	186368	2566.00	4701.00	4040.00
2	DOE/SC/Oak Ridge National Laboratory United States	Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.60
3	National Supercomputing Centre in Shenzhen (NSCS) China	Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU / 2010 Dawning	120640	1271.00	2984.30	2580.00
4	GSIC Center, Tokyo Institute of Technology Japan	TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.61
5	DOE/SC/LBNL/NERSC United States	Hopper - Cray XE6 12-core 2.1 GHz / 2010 Cray Inc.	153408	1054.00	1288.63	2910.00
6	Commissariat a l'Energie Atomique (CEA) France	Tera-100 - Bull bullx super-node S6010/S6030 / 2010 Bull SA	138368	1050.00	1254.55	4590.00
7	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM	122400	1042.00	1375.78	2345.50
8	National Institute for Computational Sciences/University of Tennessee United States	Kraken XT5 - Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	98928	831.70	1028.85	3090.00
9	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution / 2009 IBM	294912	825.50	1002.70	2268.00
10	DOE/NNSA/LANL/SNL United States	Cielo - Cray XE6 8-core 2.4 GHz / 2010 Cray Inc.	107152	816.60	1028.66	2950.00

FLOPS

- FLOPS
  = Floating Point number Operations Per Second
- 4700 TFLOPS
- = 4.7 PFLOPS
- = 4.7 x 10^{15} FLOPS

What is floating point number

- Double precision (64 bit)
  - 1 bit for sign.
  - 11 bits for exponent.
  - 52 bits for mantissa.

$$2^{52} \sim 10^{15.6}$$

- π in double precision floating point

$$\pi = 3.14159265358979$$

What is "operation" in FLOPS?

- Addition (subtraction) and multiplication (division)
- An example of floating point operation:
  - To calculate the floating point number of π^2

$$\pi^2 = 3.141592653589793 \times 3.141592653589793$$
$$= ?$$

What is Your FLOPS value?

- Using only a pen, paper and your brain…

$$\begin{aligned} \pi^2 = & \; 3.14159265358979 \\ & \times 3.14159265358979 \\ \hline & \; ?.?????????????? \end{aligned}$$

- How many seconds do you need?

Top 003 Brain FLOPS
---An International Contest---

#	Name	Country	FLOPS
1			
2			
3			

What is Your FLOPS value?

- How many seconds did you need?
- (For example) 1000 seconds?
- ……then, you are a 0.001 FLOPS computer.
- 1 mFLOPS (…not Mega FLOPS)

Human brain as a computer

If a person

- has 0.001 FLOPS speed,
- lives for 100 years,
- devotes the entire life (without sleep) to the floating point number operations (multiplication/addition),

then how many operations could be done in the life work?

Human brain as a computer

If a person

- has 0.001 FLOPS speed,
- lives for 100 years,
- devotes the entire life (without sleep) to the floating point number operations (multiplication/addition),

then how many operations could be done in the life work?

$$\left[100 \text{ years} = 3 \times 10^9 \text{ seconds}\right] \times 0.001$$

Only 3 million operations.

How about the speed of PCs?

- Example: Leibniz formula for pi.

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = \frac{\pi}{4}$$

leibniz.f90

```
program leibniz
  implicit none
  integer, parameter :: SP = kind(1.0)
  integer, parameter :: DP = selected_real_kind(2*precision(1.0_SP))
  real(DP) :: sign = -1.0_DP
  real(DP) :: sum  = 0.0_DP
  integer  :: odd_int

  do odd_int =  2*(10**6)+1 , 1 , -2
     sign = sign * (-1.0_DP)
     sum = sum + sign/odd_int
  end do

  print *, ' 4*sum = ', 4*sum
end program leibniz
```

Code:

- src/CountFlops/

leibniz.f90

```
program leibniz
  implicit none
  integer, parameter :: SP = kind(1.0)
  integer, parameter :: DP = selected_real_kind(2*precision(1.0_SP))
  real(DP) :: sign = -1.0_DP
  real(DP) :: sum  = 0.0_DP
  integer  :: i

  do i=  2*(10**6)+1 , 1 , -2
     sign = sign * (-1.0_DP)
     sum = sum + sign/i
  end do

  print *, ' 4*sum = ', 4*sum
end program leibniz
```

Repeat for one million times

Odd integers.

multiplication (once)

division (once)

addition (once)

3M Floating Point Operations

- 3 million floating point operations experiment
  - one million time steps.
  - 3 floating point number operations per each step.
- A human brain takes 100 years.
- Let's try.

Result

- 3 million floating point operations experiment
  - one million steps.
  - 3 floating point number operations per each step.
- A human brain takes 100 years.
- … 0.009 second by this PC.
- $3 \times 10^{6} / 9 \times 10^{-3}$ FLOPS
- = 0.3 GFLOPS

Speed contrast

- Our brain = mFLOPS (10^{-3} FLOPS)
- PC = GFLOPS (10^{9} FLOPS)
- Supercomputer = 10^{15} FLOPS

Concorde is only 400-500 times faster than our walk speed.

Speed Contrast

- Speed contrast: Supercomputer / PC = 10^6
- Speed contrast: Supercomputer / brain = 10^{18}
  - Human walk = 1-2 m/s
  - Speed of light = 3×10^8 m/s
  - Speed contrast: light / walk = 10^8

The approach of computer simulation

- Your problem (to be solved)
- Convert (or approximate) it into the form with +, -, *, /
- Make computers solve it.

Methods used in MHD simulations

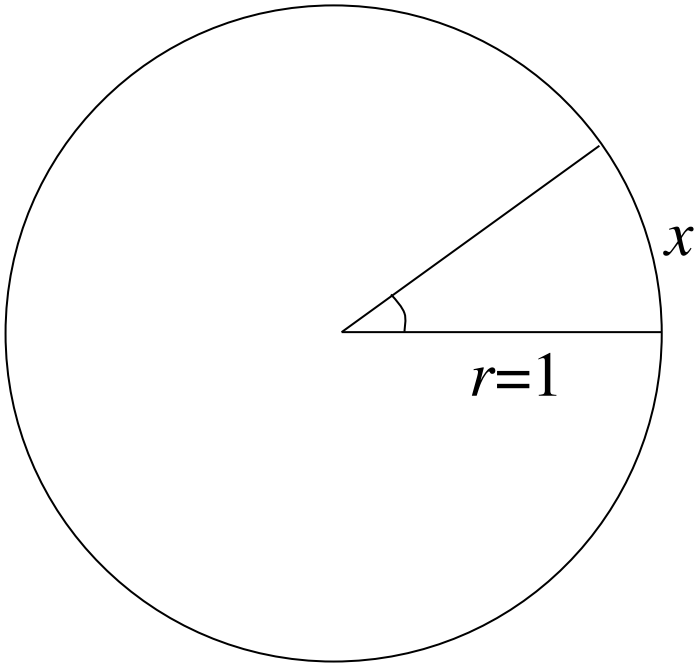
- Spectral methods
- Finite difference method
- Finite volume method
- Finite element method
- • • •

Discretization

- NS eq., MHD equations., etc…
- Continuous x (space) and t (time)
- Differential equations
- Convert the differential Equations
  ➔ Eqs with +, -, *, /
- “Discretization”

Various numerical method:
An explanation through a simple 1-D problem

The diffusion equation on a circle

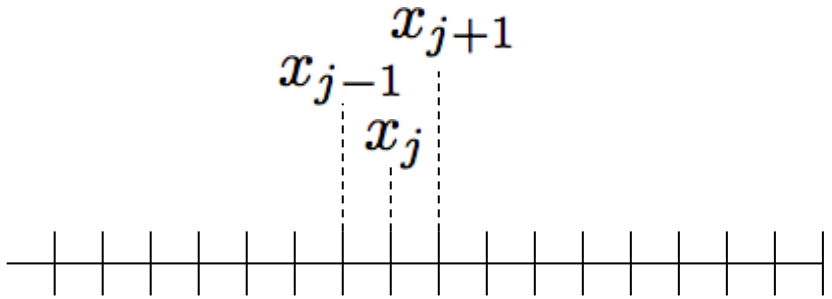


$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$0 \leq x < 2\pi$$

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

(1) FDM: Finite Difference Method



$$\frac{d\psi}{dx} \sim \frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x}$$

$$\frac{d^2\psi}{dx^2} \sim \frac{\psi_{j+1} - 2\psi_j + \psi_{j+1}}{\Delta x}$$

$$\frac{d\psi_j}{dt} = \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

(1) FDM: Finite Difference Method

$$\frac{d\psi_j}{dt} = \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

$$\frac{d\psi_j}{dt} = f(\psi_1, \psi_2, \cdots, \psi_N)$$

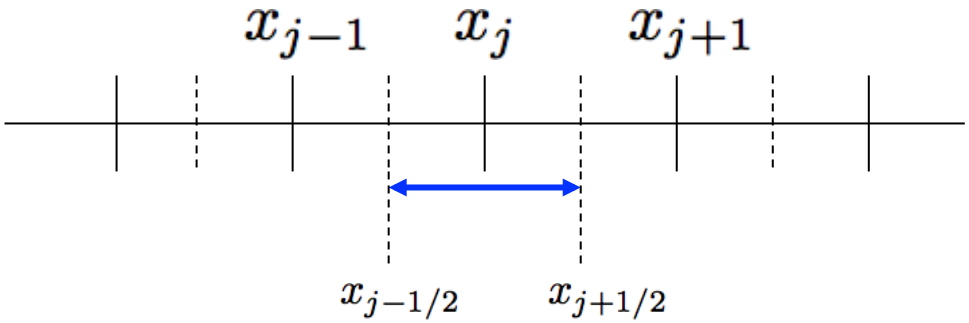
==> Common numerical integrators (e.g., Runge-Kutta).

We will see in practice by this note PC.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

(2) FVM: Finite Volume Method

$$\frac{\partial \psi}{\partial t} = \frac{\partial F}{\partial x}, \qquad F = \frac{\partial \psi}{\partial x}$$



$$\int_{x_{j-1/2}}^{x_{j+1/2}} \frac{\partial \psi}{\partial t} dx = \int_{x_{j-1/2}}^{x_{j+1/2}} \frac{\partial F}{\partial x} dx,$$

cell average $\left(\bar{f} \equiv \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} f dx \right)$

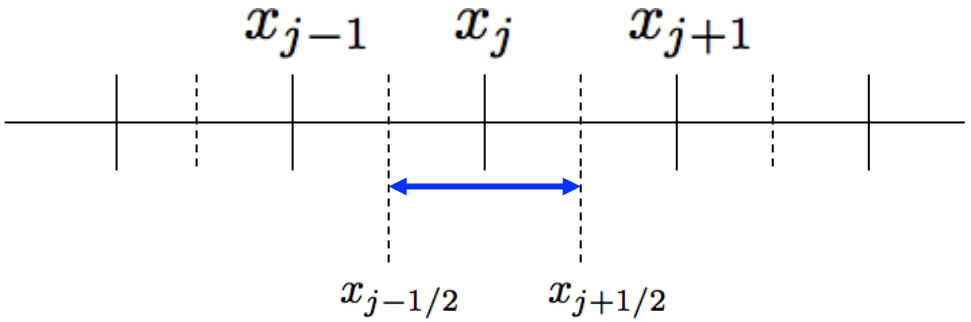
$$\frac{d\bar{\psi}_j}{dt} = \left[F(x_{j+1/2}) - F(x_{j-1/2}) \right] / \Delta x$$

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

(2) FVM: Finite Volume Method

$$\frac{d\bar{\psi}_j}{dt} = \left[F(x_{j+1/2}) - F(x_{j-1/2})\right] / \Delta x$$

$$F = \frac{\partial \psi}{\partial x}$$



$$F(x_{j+1}) = \frac{\bar{\psi}(j+1) - \bar{\psi}(j-1)}{\Delta x}$$

$$\Rightarrow \quad \frac{d\bar{\psi}_j}{dt} = \frac{\bar{\psi}_{j+1} - 2\bar{\psi}_j + \bar{\psi}_{j-1}}{(\Delta x)^2}$$

Same as FDM.

$$\boxed{\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}}$$

(3) FEM for 1D diffusion eq.

$\forall a(x)$, (periodic)

$$\int_{-\pi}^{\pi} a(x) \frac{\partial \psi(x)}{\partial x} dx = \int_{-\pi}^{\pi} a(x) \frac{\partial^2 \psi(x)}{\partial x^2} dx$$

$$\boxed{\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}}$$

FEM for 1D diffusion eq.

$\forall a(x)$, (periodic)

$$\int_{-\pi}^{\pi} a(x) \frac{\partial \psi(x)}{\partial x} dx = \int_{-\pi}^{\pi} a(x) \frac{\partial^2 \psi(x)}{\partial x^2} dx$$

$$= \left[a \frac{d\psi}{\partial x} \right]_{-\pi}^{\pi} - \int_{-\pi}^{\pi} \frac{da(x)}{dx} \frac{\partial \psi}{\partial x} dx$$

$$\boxed{\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}}$$

FEM for 1D diffusion eq.

$\forall a(x)$, (periodic)

$$\int_{-\pi}^{\pi} a(x) \frac{\partial \psi(x)}{\partial x} dx = \int_{-\pi}^{\pi} a(x) \frac{\partial^2 \psi(x)}{\partial x^2} dx$$

$$= \left[a \frac{d\psi}{\partial x} \right]_{-\pi}^{\pi} - \int_{-\pi}^{\pi} \frac{da(x)}{dx} \frac{\partial \psi}{\partial x} dx$$

$$= - \int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$

FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

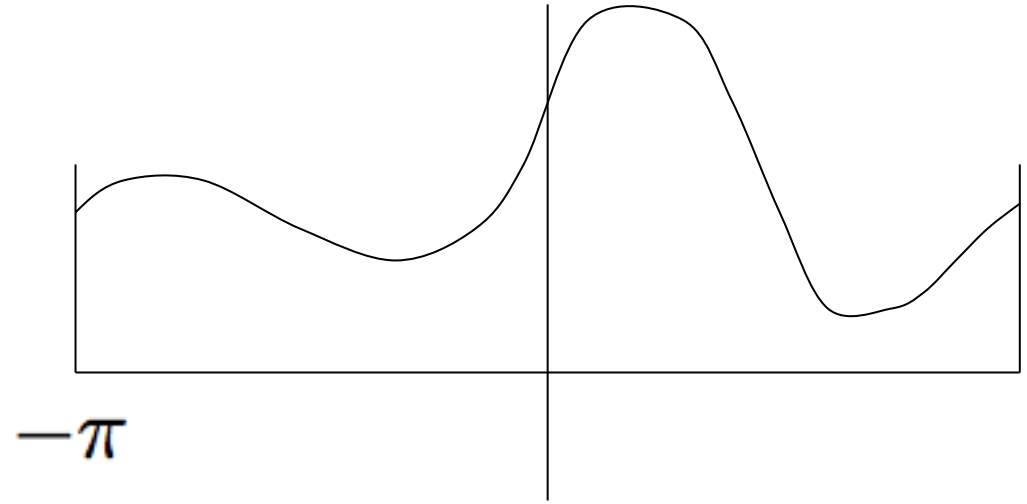
$$\forall a(x), \qquad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = - \int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$

FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

equivalent

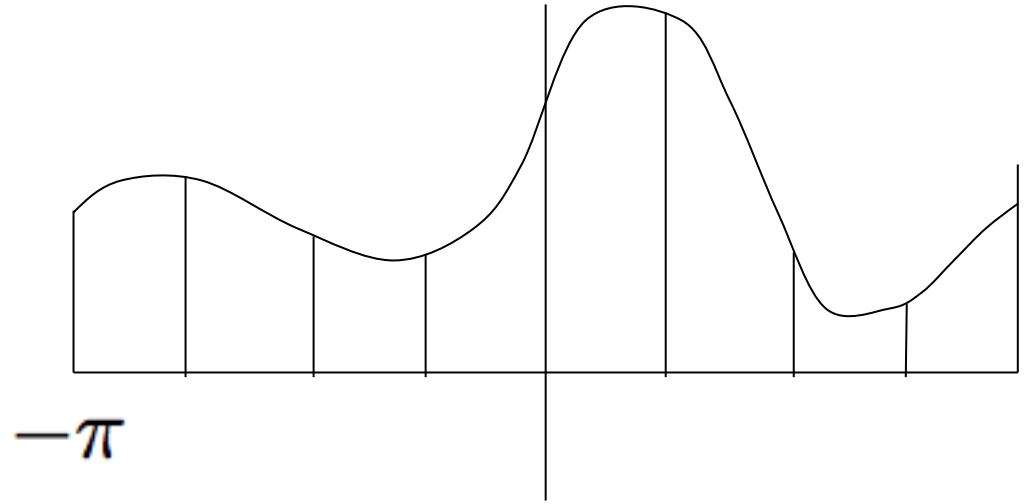
$$\forall a(x), \quad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = -\int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$



FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$\forall a(x), \quad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = -\int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$

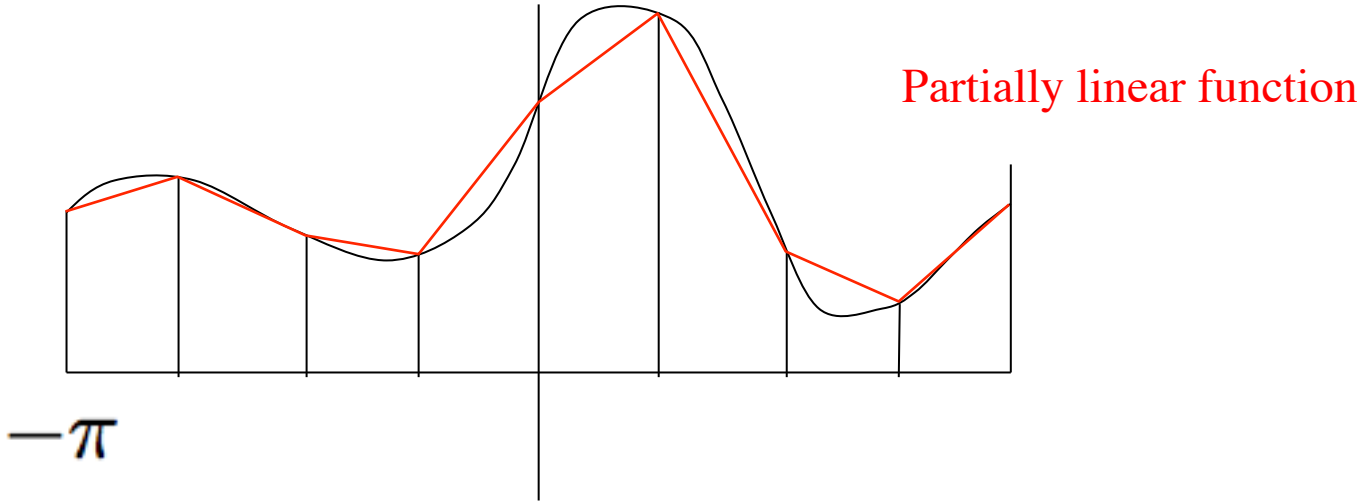


Divide the space x into finite number of sub-spaces (finite element)

FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$\forall a(x), \quad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = -\int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$

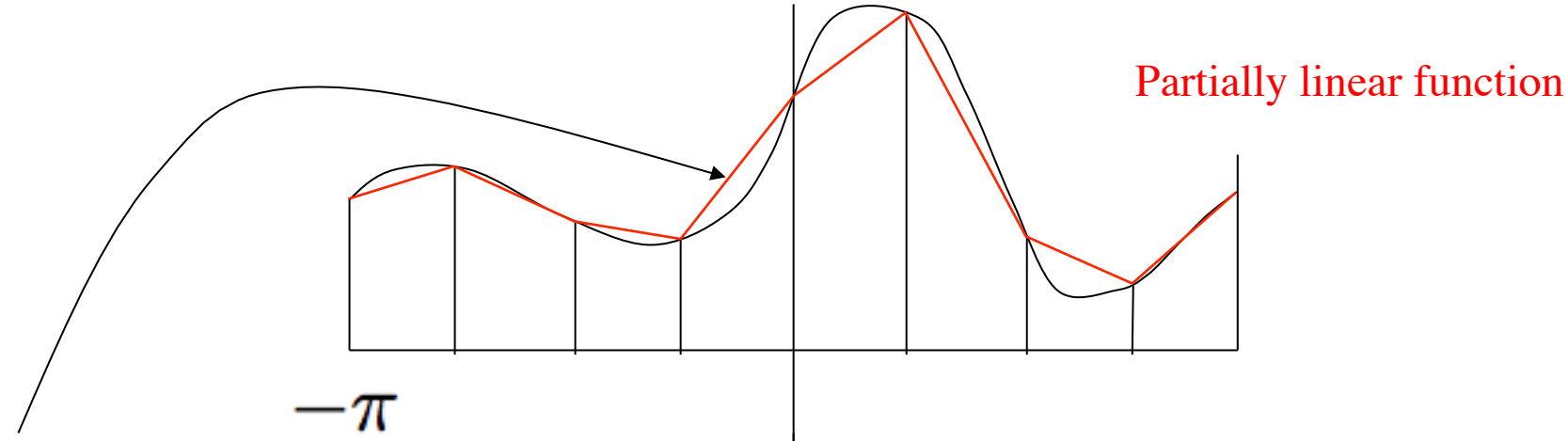


Approximate the function by a partially linear function.

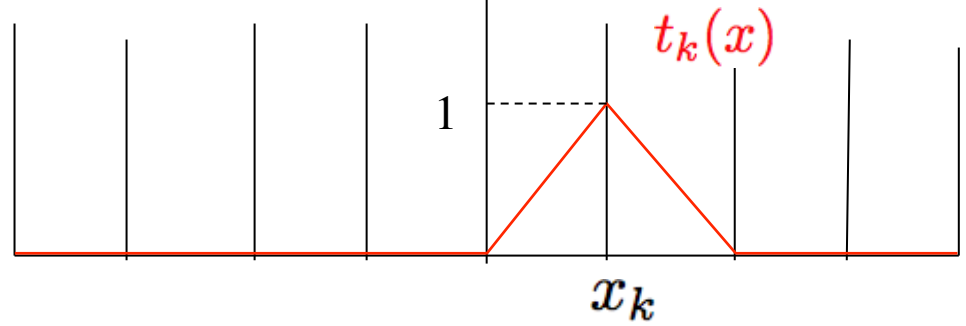
FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$\forall a(x), \qquad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = - \int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$



This profile is expressed by a linear combination of these delta-like (triangle) functions.



$$\psi(x,t) = \sum_k \psi_k(t)\, t_k(x)$$

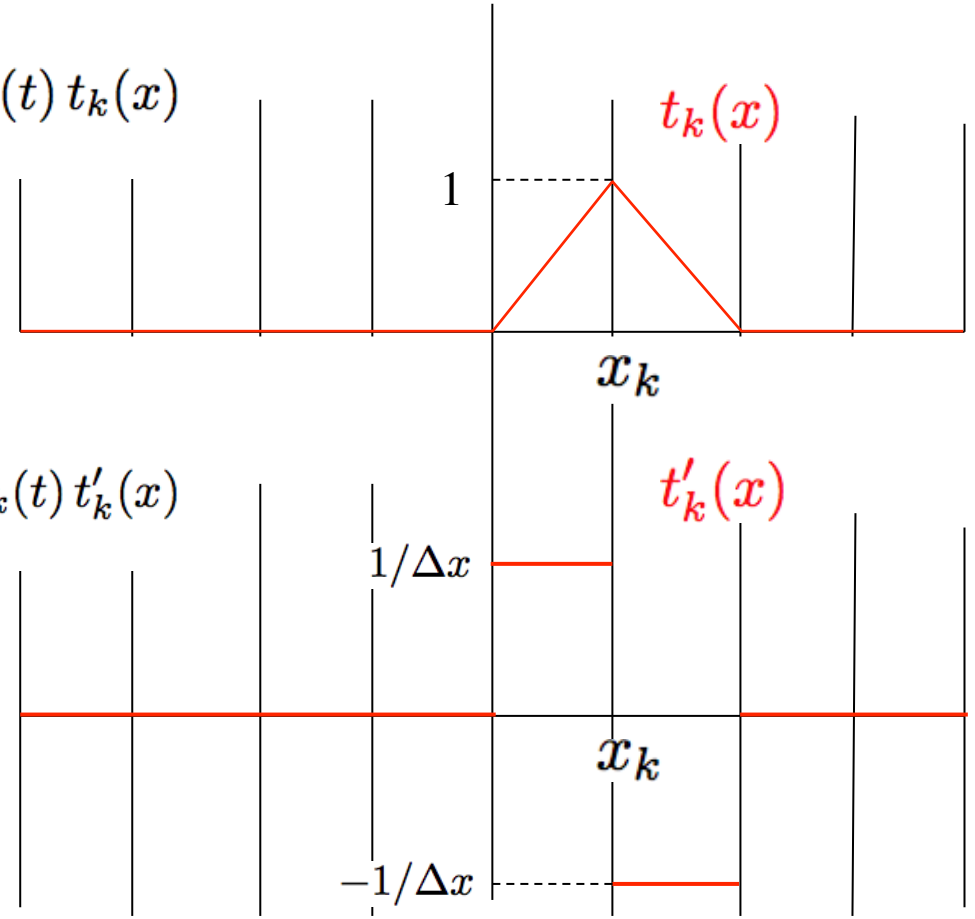
FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$\forall a(x), \qquad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = - \int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$

$$\psi(x,t) = \sum_k \psi_k(t)\, t_k(x)$$

$$\frac{\partial \psi(x,t)}{\partial x} = \sum_k \psi_k(t)\, t_k'(x)$$



FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$\forall a(x), \qquad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = -\int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$

Substituting $\psi(x,t) = \sum_k \psi_k(t)\, t_k(x)$ and $\frac{\partial \psi(x,t)}{\partial x} = \sum_k \psi_k(t)\, t_k'(x)$

$$\sum_k \frac{d\psi_k(t)}{dt} \int_{-\pi}^{\pi} a(x) t_k(x) dx = -\sum_k \psi_k(t) \int_{-\pi}^{\pi} a'(x)\, t_k'(x) dx$$

FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

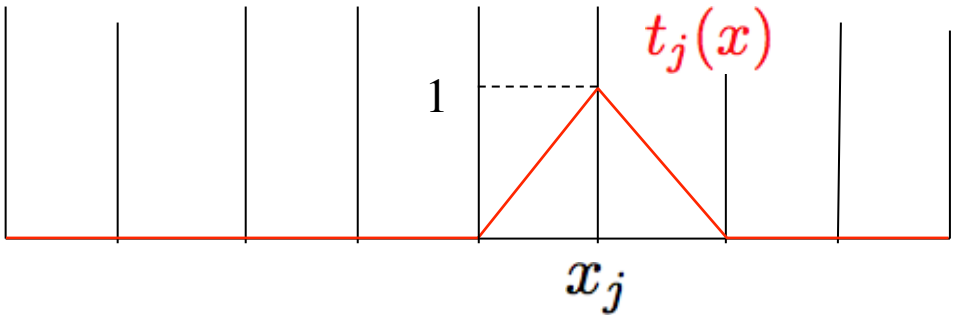
$$\forall a(x), \qquad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = -\int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$

Substituting $\psi(x,t) = \sum_k \psi_k(t)\, t_k(x)$ and $\frac{\partial \psi(x,t)}{\partial x} = \sum_k \psi_k(t)\, t_k'(x)$

$$\sum_k \frac{d\psi_k(t)}{dt} \int_{-\pi}^{\pi} a(x) t_k(x) dx = -\sum_k \psi_k(t) \int_{-\pi}^{\pi} a'(x)\, t_k'(x) dx$$

Take $a(x) = t_j(x), \quad (j = 1, 2, 3, \ldots)$

$a'(x) = t_j'(x), \quad (j = 1, 2, 3, \ldots)$



FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$\forall a(x), \qquad \int_{-\pi}^{\pi} a(x) \frac{\partial \psi}{\partial t} dx = -\int_{-\pi}^{\pi} a'(x) \frac{\partial \psi}{\partial x} dx$$

Substituting $\psi(x,t) = \sum_k \psi_k(t)\, t_k(x)$ and $\frac{\partial \psi(x,t)}{\partial x} = \sum_k \psi_k(t)\, t_k'(x)$

$$\sum_k \frac{d\psi_k(t)}{dt} \int_{-\pi}^{\pi} a(x) t_k(x) dx = -\sum_k \psi_k(t) \int_{-\pi}^{\pi} a'(x)\, t_k'(x) dx$$

Take $a(x) = t_j(x), \quad (j = 1, 2, 3, \ldots)$

$a'(x) = t_j'(x), \quad (j = 1, 2, 3, \ldots)$

$$\sum_k \frac{d\psi_k(t)}{dt} \underbrace{\int_{-\pi}^{\pi} t_j(x) t_k(x) dx}_{P_{jk}} = -\sum_k \psi_k(t) \underbrace{\int_{-\pi}^{\pi} t_j'(x)\, t_k'(x) dx}_{Q_{jk}}$$

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

FEM for 1D diffusion eq.

$$\sum_k P_{jk} \frac{d\psi_k(t)}{dt} = -\sum_k Q_{jk} \psi_k(t)$$

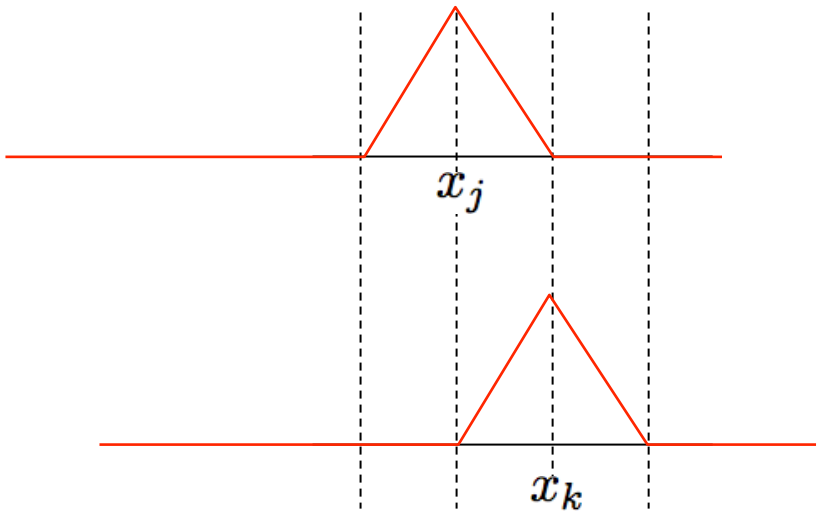
$$P_{jk} = \int_{-\pi}^{\pi} t_j(x) t_k(x) dx \qquad Q_{jk} = -\sum_k \psi_k(t) \int_{-\pi}^{\pi} t_j'(x)\, t_k'(x) dx$$

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

FEM for 1D diffusion eq.

$$\sum_k P_{jk} \frac{d\psi_k(t)}{dt} = -\sum_k Q_{jk} \psi_k(t)$$

$$P_{jk} = \int_{-\pi}^{\pi} t_j(x) t_k(x) dx = \begin{cases} \frac{2}{3}\Delta x & (j = k) \\ \frac{1}{6}\Delta x & (j = k \pm 1) \\ 0 & \text{(others)} \end{cases}$$

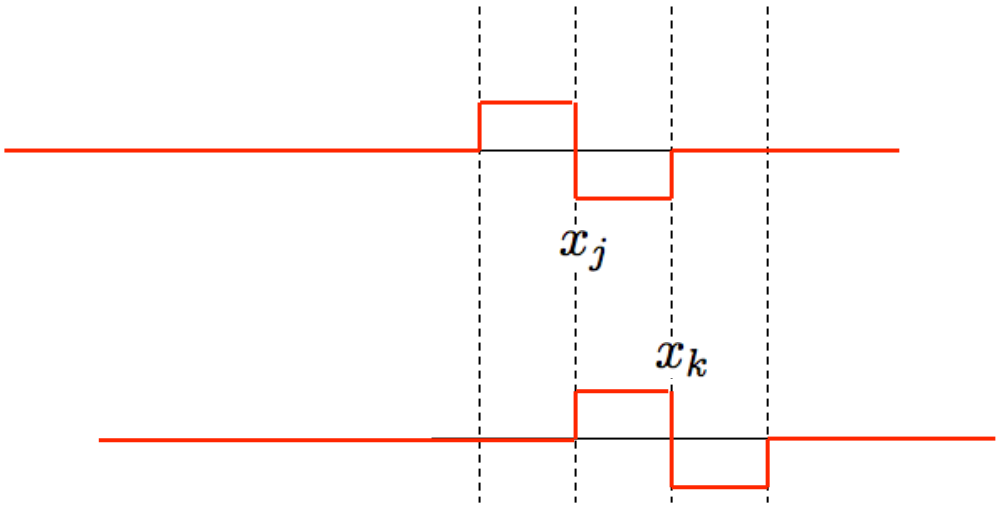


FEM for 1D diffusion eq.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

$$\sum_k P_{jk} \frac{d\psi_k(t)}{dt} = -\sum_k Q_{jk} \psi_k(t)$$

$$Q_{jk} = -\sum_k \psi_k(t) \int_{-\pi}^{\pi} t_j'(x)\, t_k'(x) dx = \begin{cases} \frac{2}{\Delta x} & (j = k) \\ -\frac{1}{\Delta x} & (j = k \pm 1) \\ 0 & (\text{others}) \end{cases}$$



$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

FEM for 1D diffusion eq.

$$\sum_k P_{jk} \frac{d\psi_k(t)}{dt} = -\sum_k Q_{jk} \psi_k(t)$$

Then this leads to;

$$\frac{1}{6}\left(\frac{d\psi_{j-1}}{dt} + 4\frac{d\psi_j}{dt} + \frac{d\psi_{j+1}}{dt}\right) = \frac{\psi_{j-1} - 2\psi_j + \psi_{j+1}}{(\Delta x)^2}$$

for *j=2, …, N-1*

And with the boundary condition:

$$\frac{d\psi_1}{dt} - \frac{d\psi_{N-1}}{dt} = 0$$

$$\frac{d\psi_N}{dt} - \frac{d\psi_2}{dt} = 0$$

We get a matrix equation as…

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

FEM for 1D diffusion eq.

$$A\,\mathbf{x} = B\,\mathbf{y}$$

where (for example, N=10 case)

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad B = \frac{6}{(\Delta x)^2} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} \frac{d\psi_1}{dt} \\ \frac{d\psi_2}{dt} \\ \frac{d\psi_3}{dt} \\ \vdots \\ \frac{d\psi_N}{dt} \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \vdots \\ \psi_N \end{pmatrix}$$

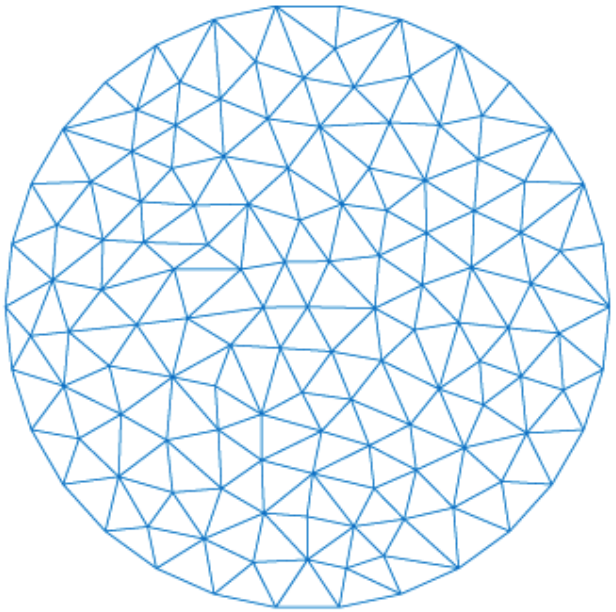
The final step is to solve this matrix eq.

The method depends on how we approximate the time derivative $\frac{d\psi_j}{dt}$

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

(3) FEM: Finite Element Method

$$\frac{1}{6}\left(\frac{\psi_{j-1}}{dt} + 4\frac{d\psi_j}{dt} + \frac{\psi_{j+1}}{dt}\right) = \frac{\psi_{j-1} - 2\psi_j + \psi_{j+1}}{(\Delta x)^2}$$



2D example: Flexible mesh design.

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

(4) Spectral Method

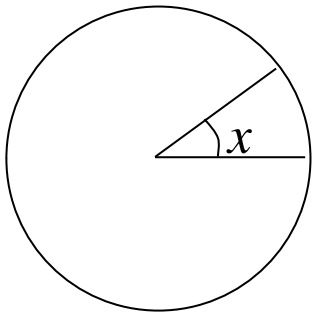
Expansion by a set of orthonormal basis.

$$\psi(x,t) = \sum_k \psi_k(t) a_k(x)$$

$$(a_j, a_k) \equiv \int_{-\pi}^{\pi} a_j^\dagger(x)\, a_k(x) dx = \delta_{jk}$$

Example: Fourier functions.

$$a_k(x) = \frac{\exp(ikx)}{\sqrt{2\pi}}$$



$$\psi(x,t) = \sum_m \psi_m(t) \frac{\exp(imx)}{\sqrt{2\pi}}$$

$$\Rightarrow \quad \frac{\partial^2 \psi(x,t)}{\partial x^2} = -\sum_m m^2\, \psi_m(t) \frac{\exp(imx)}{\sqrt{2\pi}}$$

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

(4) Spectral Method

$$(a_m, \frac{\partial \psi}{\partial t}) = (a_m, \frac{\partial^2 \psi}{\partial x^2})$$

$$\psi(x,t) = \sum_k \psi_k(t)\, a_k(x)$$

$$\frac{\partial^2 \psi(x,t)}{\partial x^2} = -\sum_k k^2\, \psi_k(t)\, a_k(x)$$

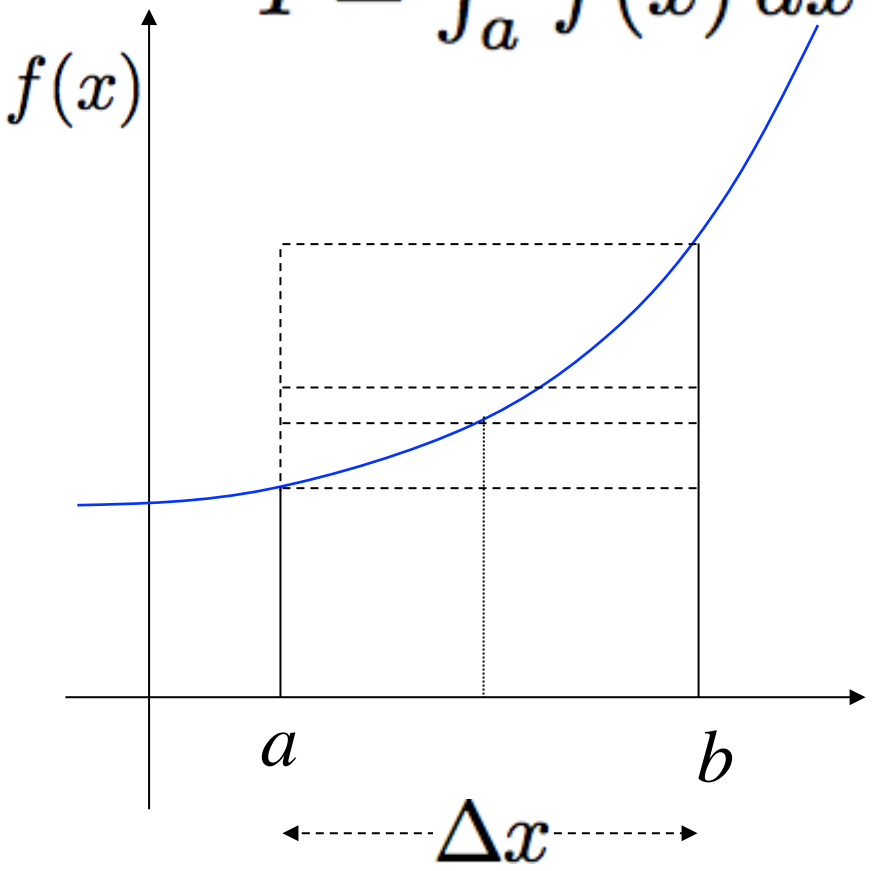
$$(a_j, a_k) = \delta_{jk}$$

$$\frac{d\psi_m(t)}{dt} = -m^2\, \psi_m(t)$$

Numerical integration

Numerical integration

$$I = \int_a^b f(x)\,dx$$



1) $I = \Delta x\, f(a)$

Error $\propto O(\Delta x^2)$

2) Trapezoid rule

$$I = \frac{\Delta x}{2}\,[f(a) + f(b)]$$

Error $\propto O(\Delta x^3)$

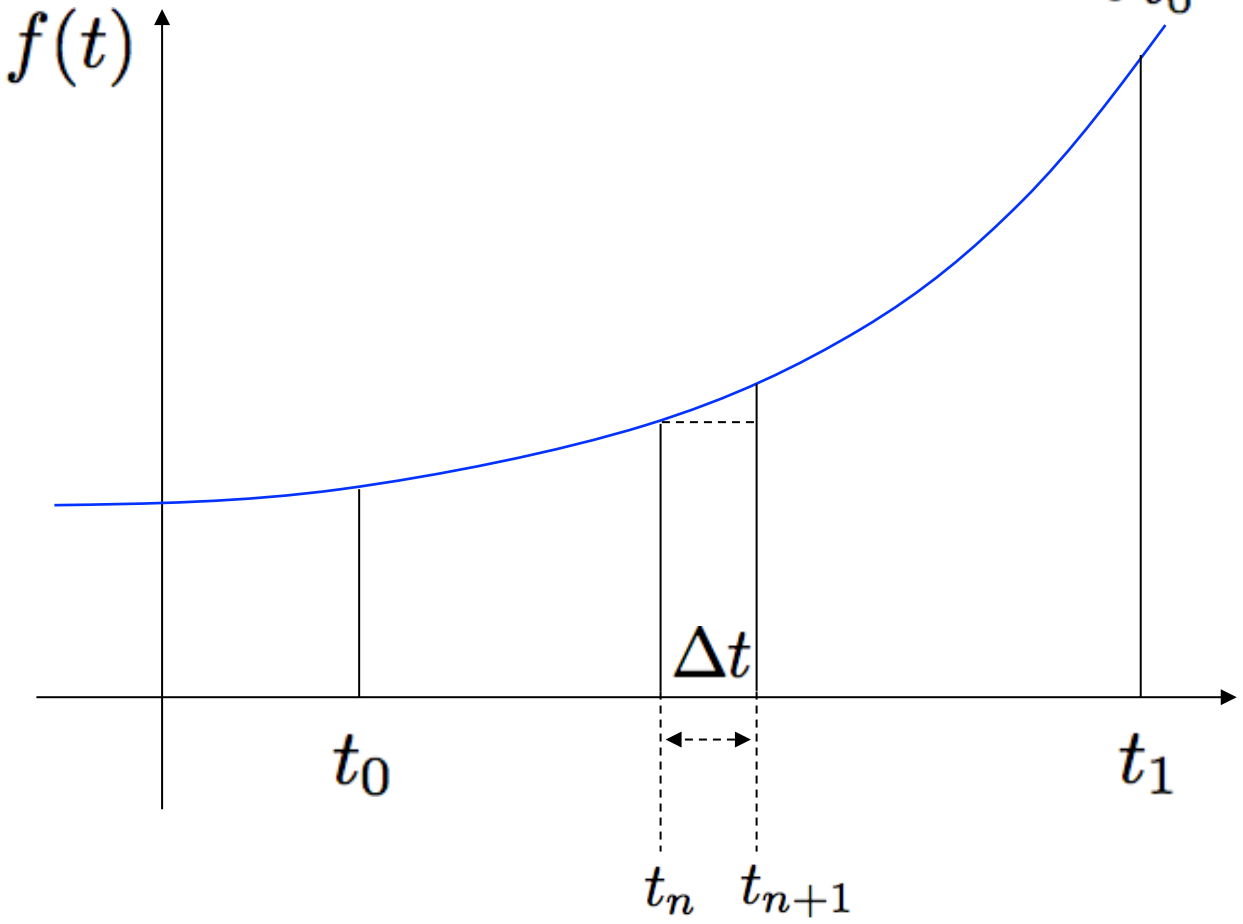
3) Simpson's rule

$$I = \frac{\Delta x}{6}\,\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Error $\propto O(\Delta x^5)$

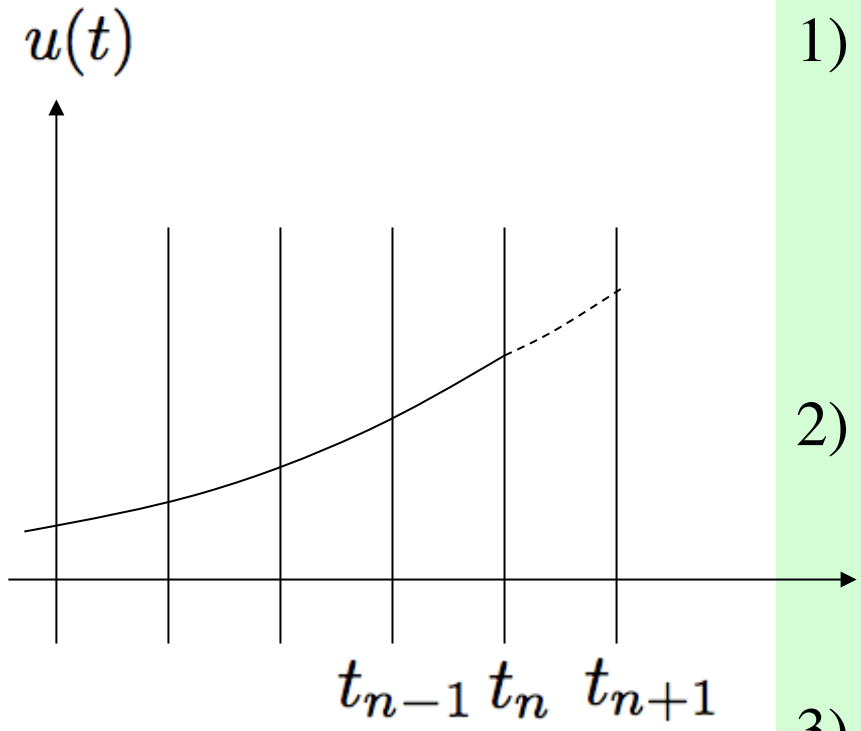
$$\frac{du(t)}{dt} = f(u(t), t)$$

$$u(t_1) = u(t_0) + \int_{t_0}^{t_1} f\, dt$$



$$\frac{du(t)}{dt} = f(u(t), t)$$

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f\, dt$$



1)

$$u(t_{n+1}) = u(t_n) + \Delta t\, f(t_n)$$

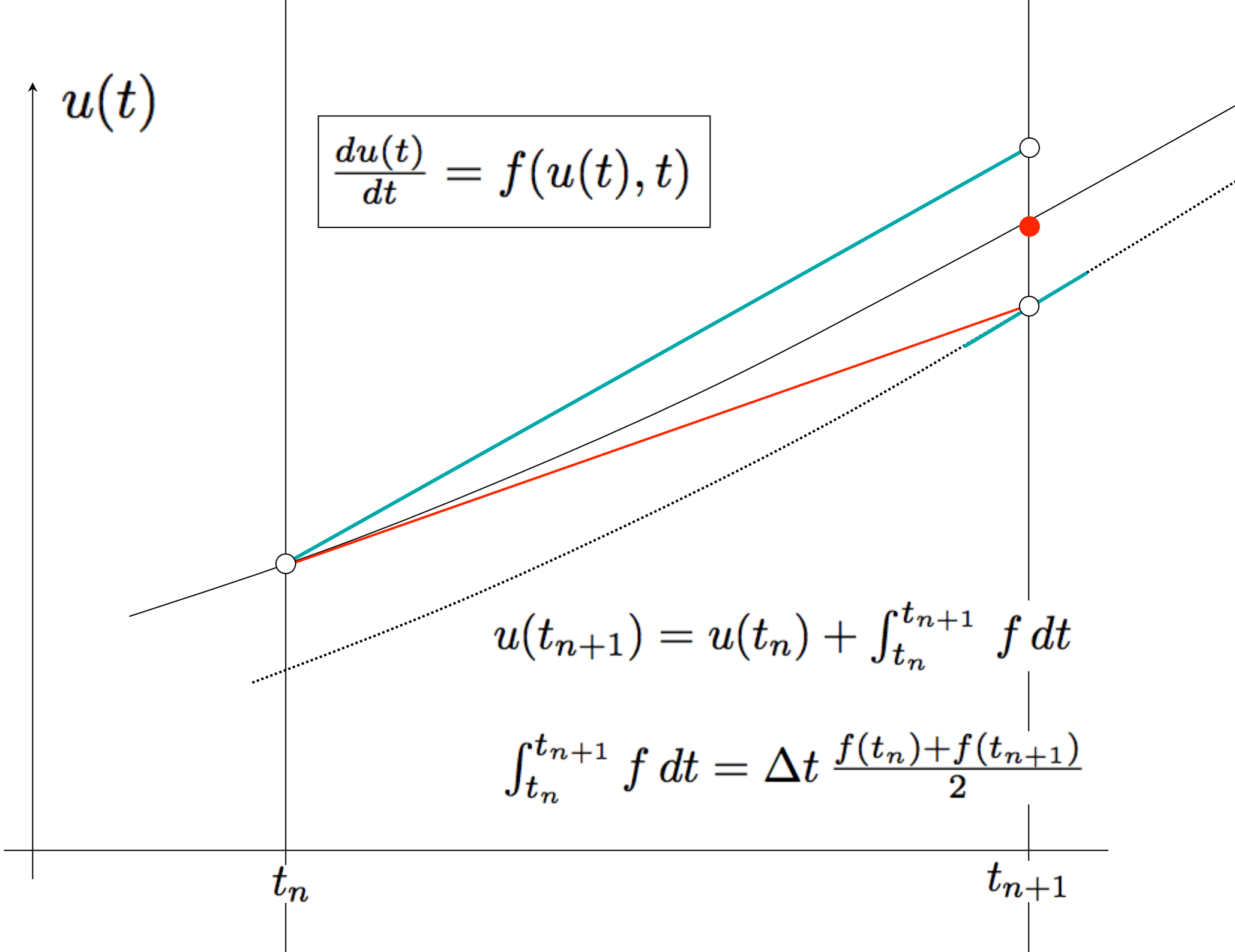
⇨ 1st order Euler method

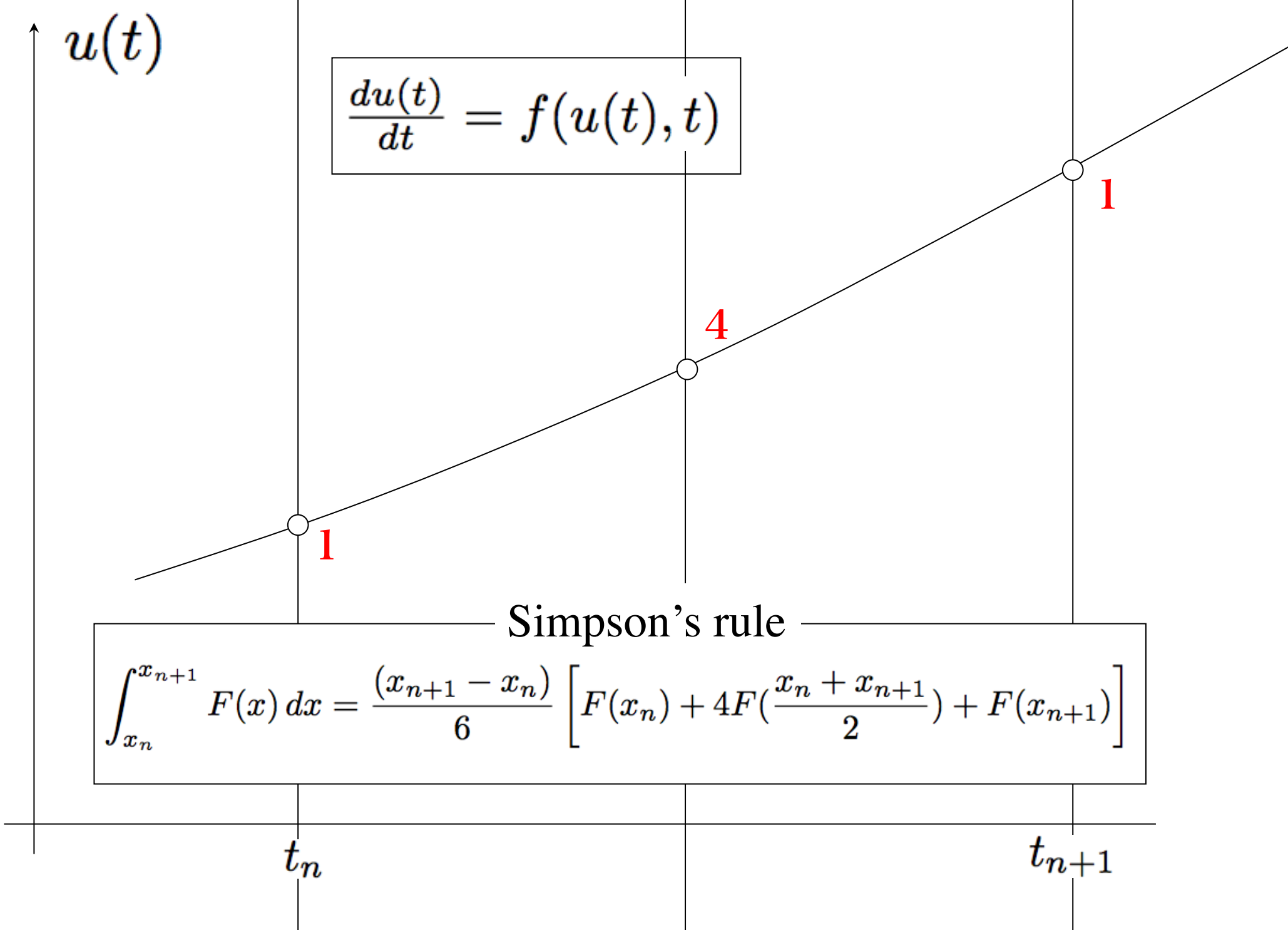
2) Trapezoid rule

⇨ 2nd order Runge-Kutta method

3) Simpson's rule

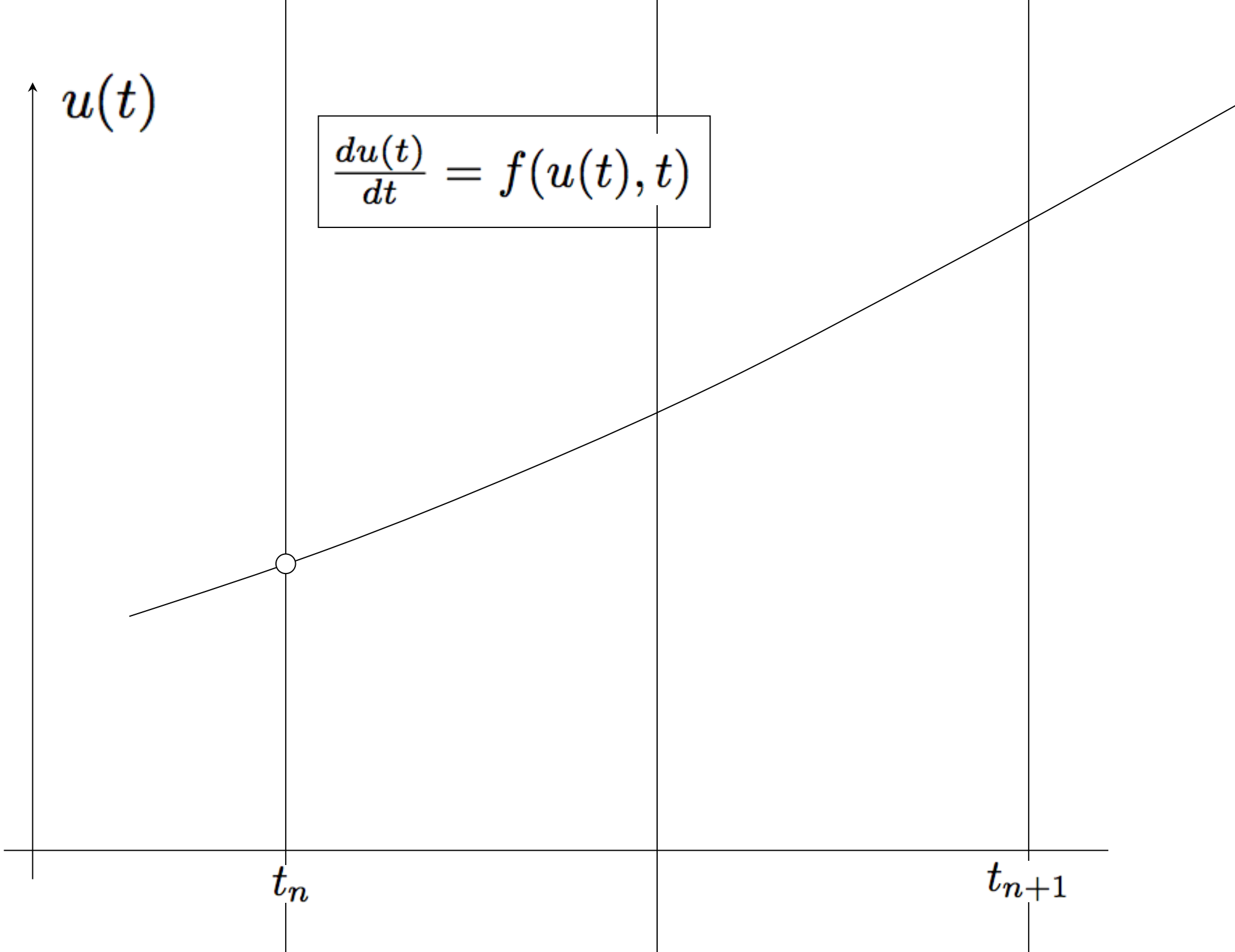
⇨ 4th order Runge-Kutta method

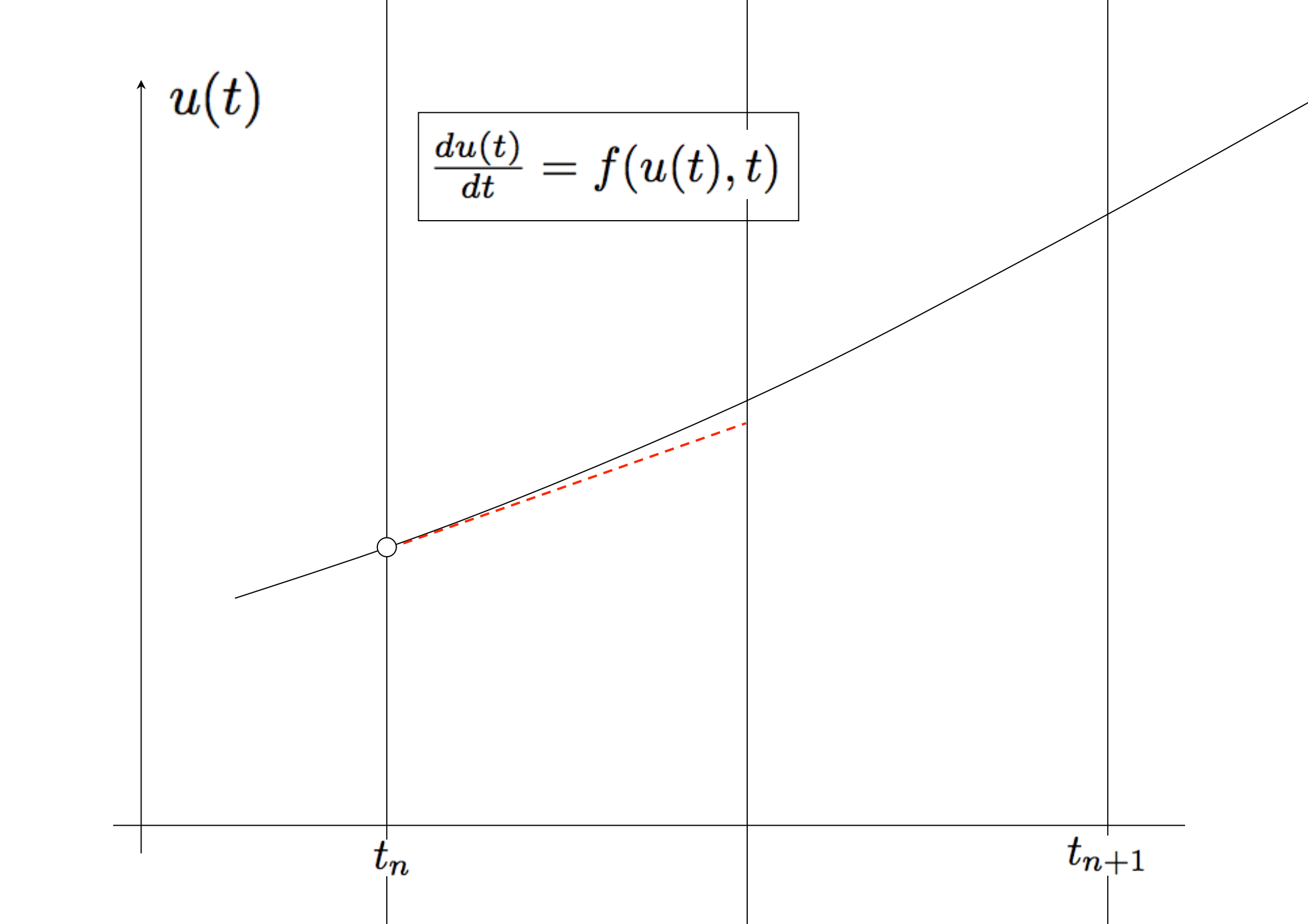

$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f\, dt$
$\int_{t_n}^{t_{n+1}} f\, dt = \Delta t\, \frac{f(t_n)+f(t_{n+1})}{2}$
t_n
t_{n+1}

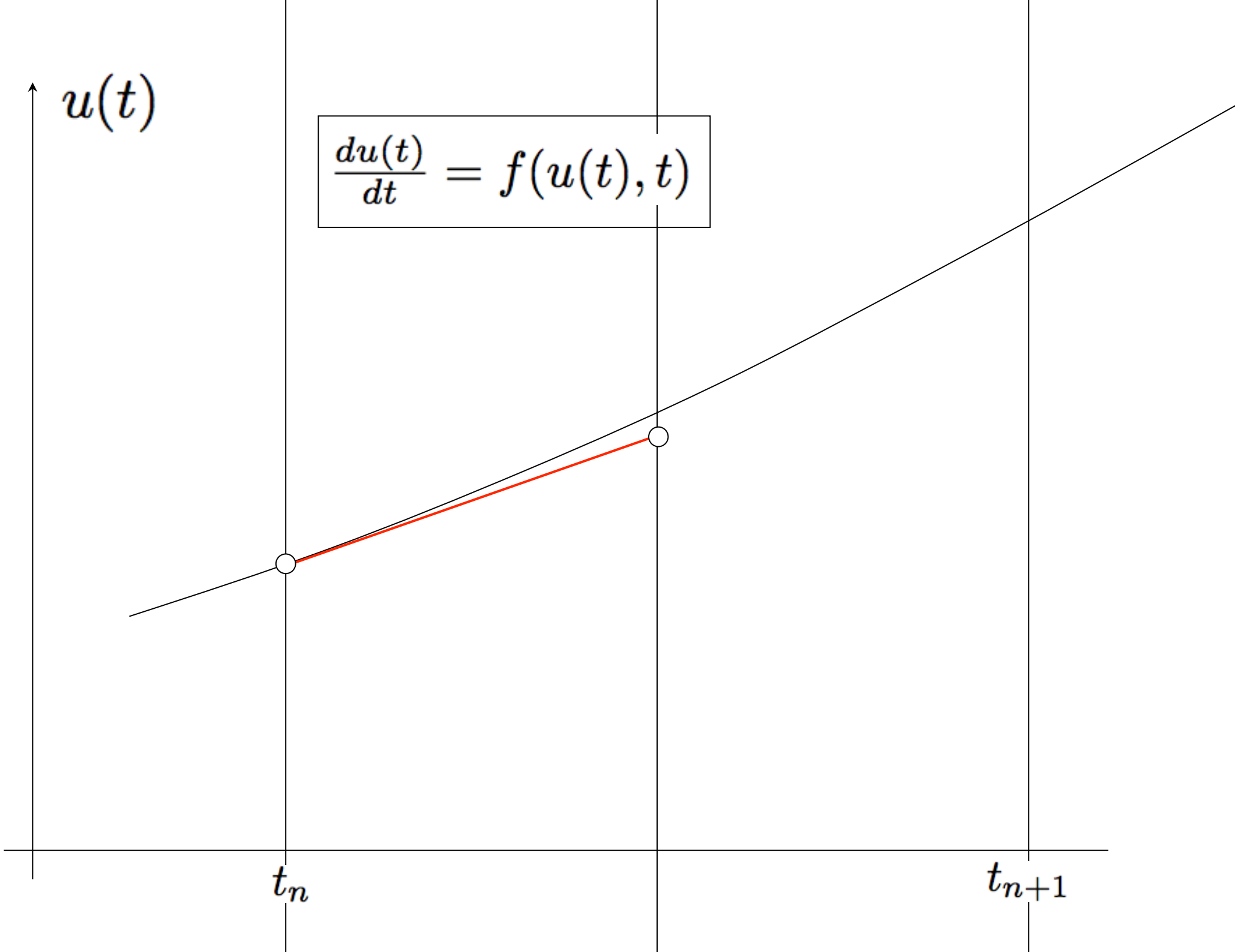


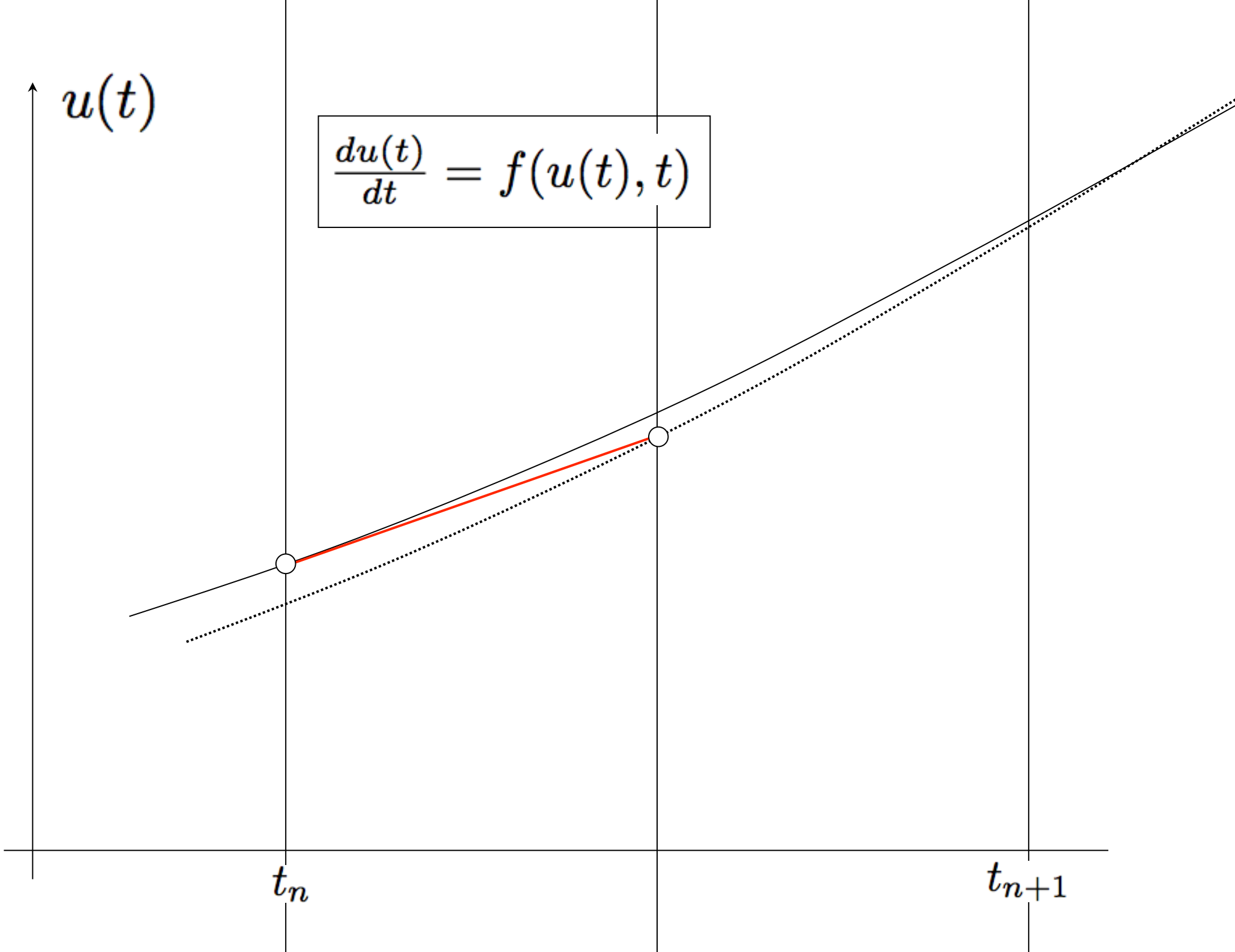
Simpson's rule

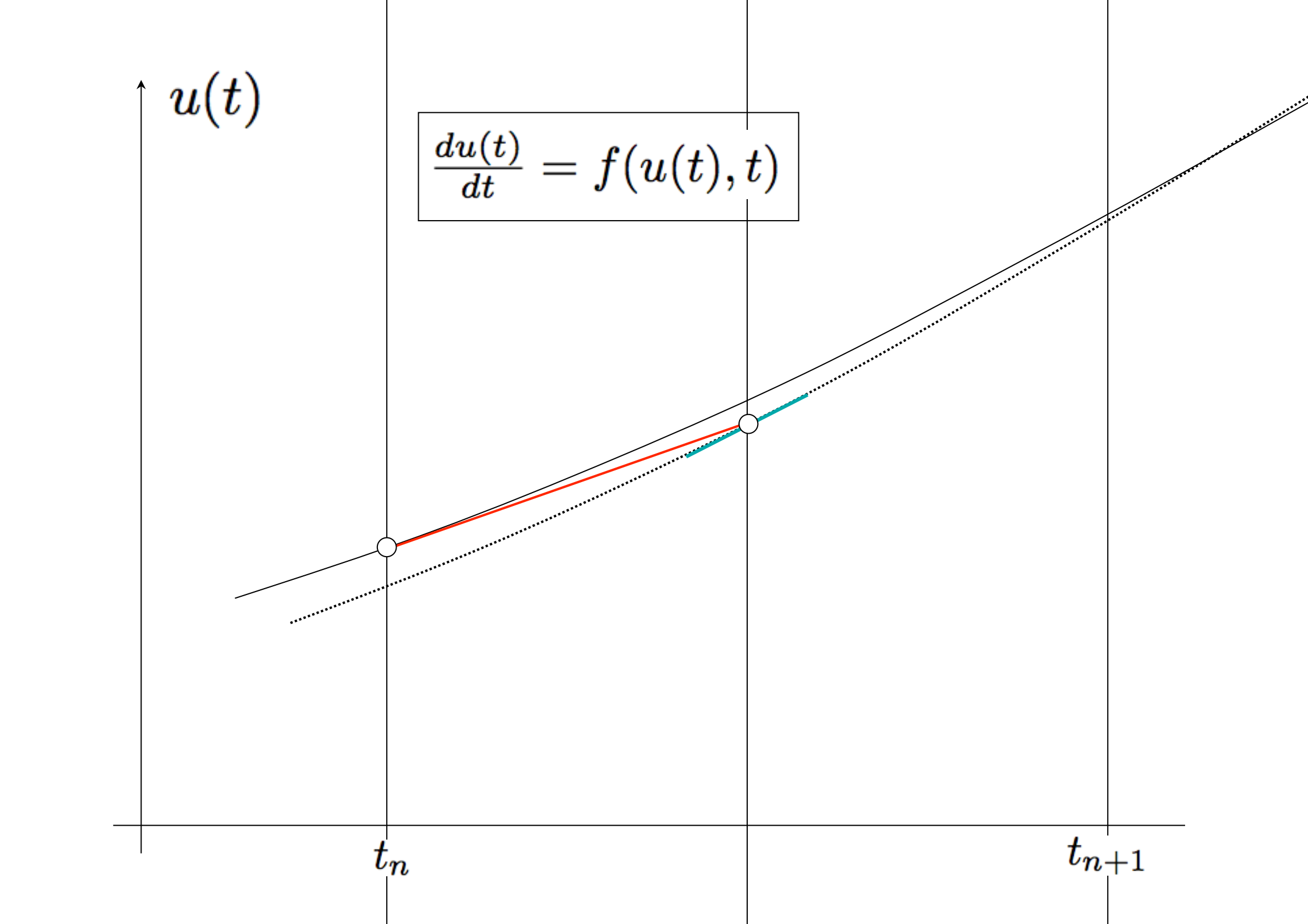
$$\int_{x_n}^{x_{n+1}} F(x)\,dx = \frac{(x_{n+1} - x_n)}{6}\left[F(x_n) + 4F(\frac{x_n + x_{n+1}}{2}) + F(x_{n+1})\right]$$

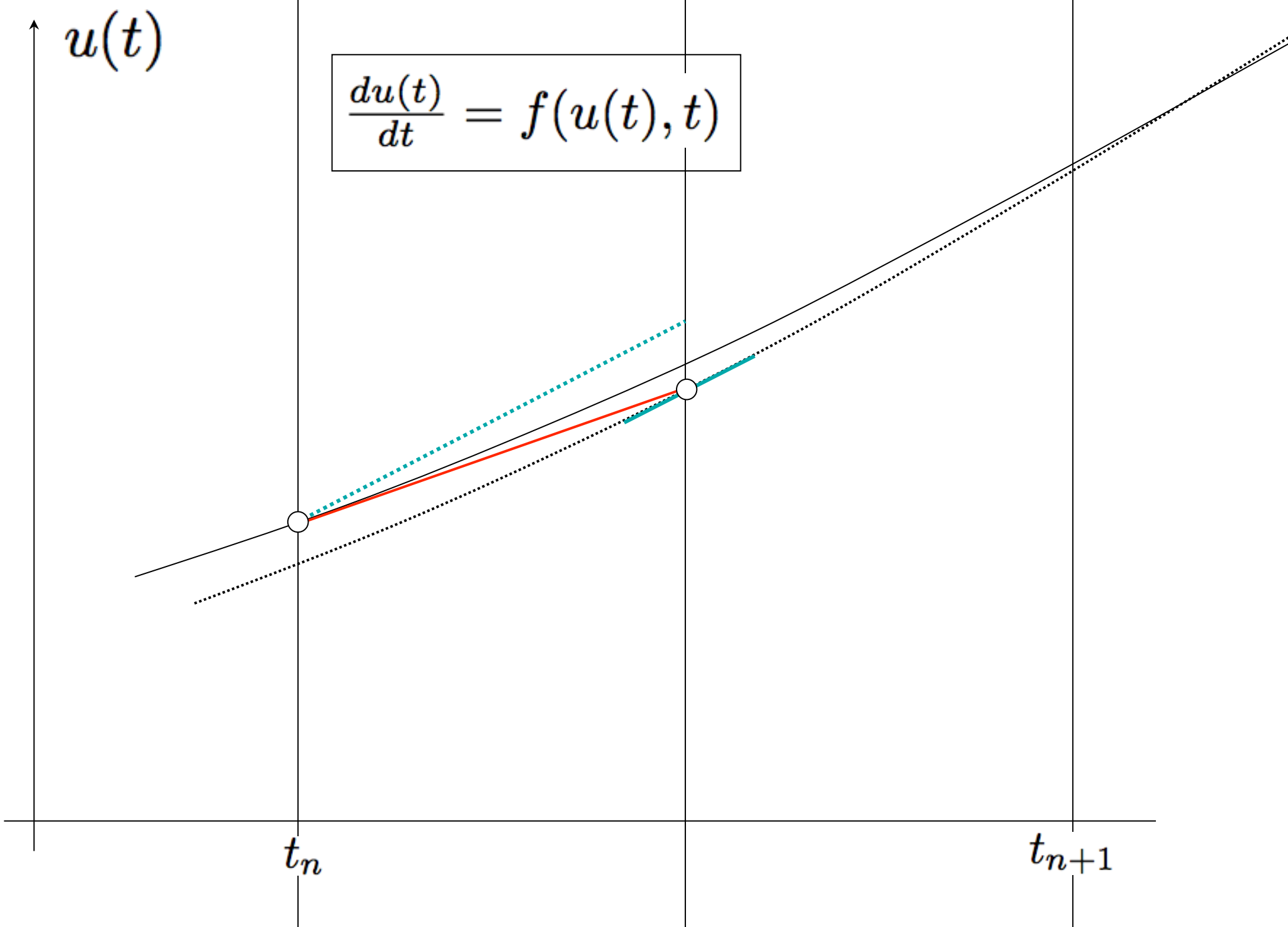

$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}

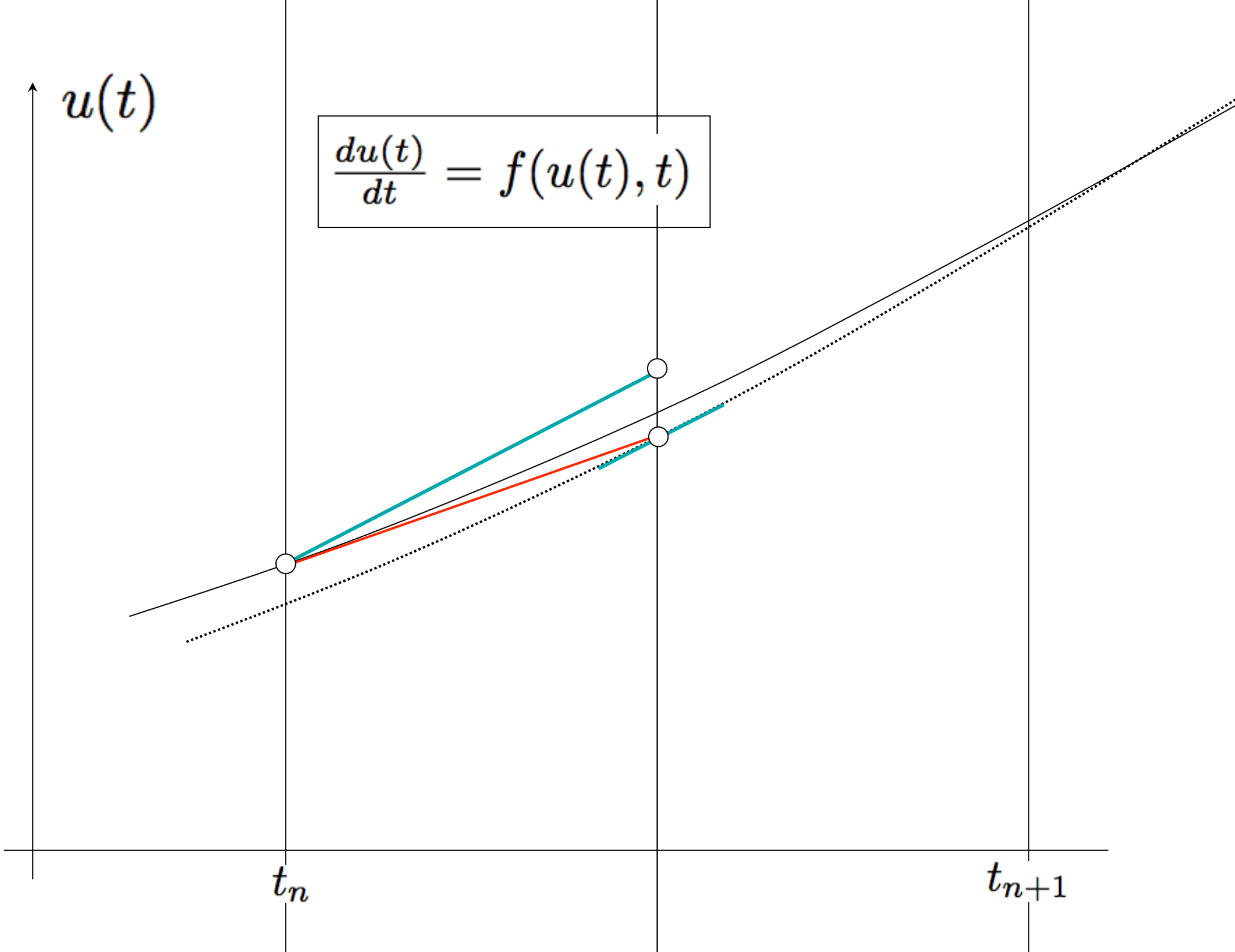

$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}

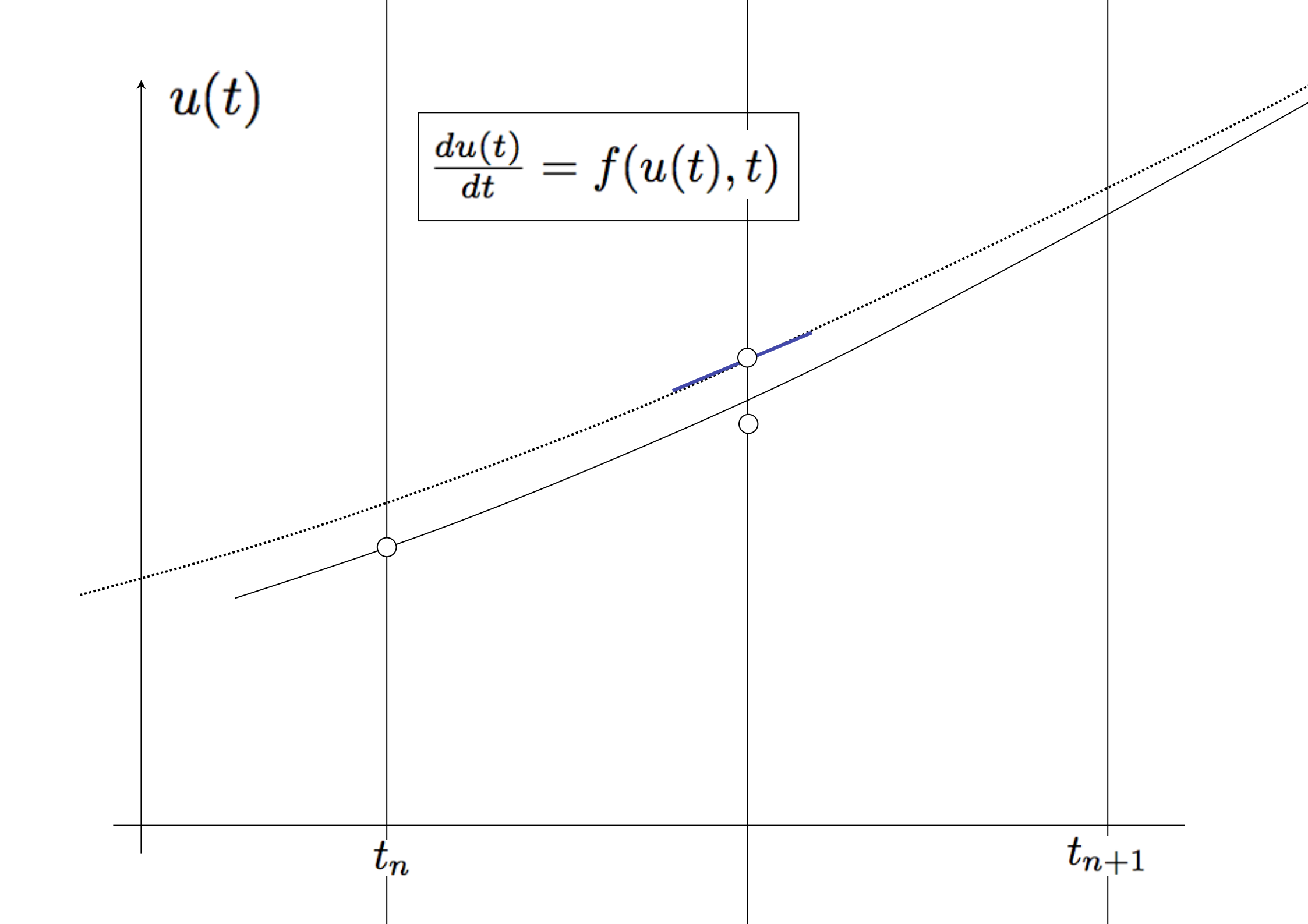

$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}

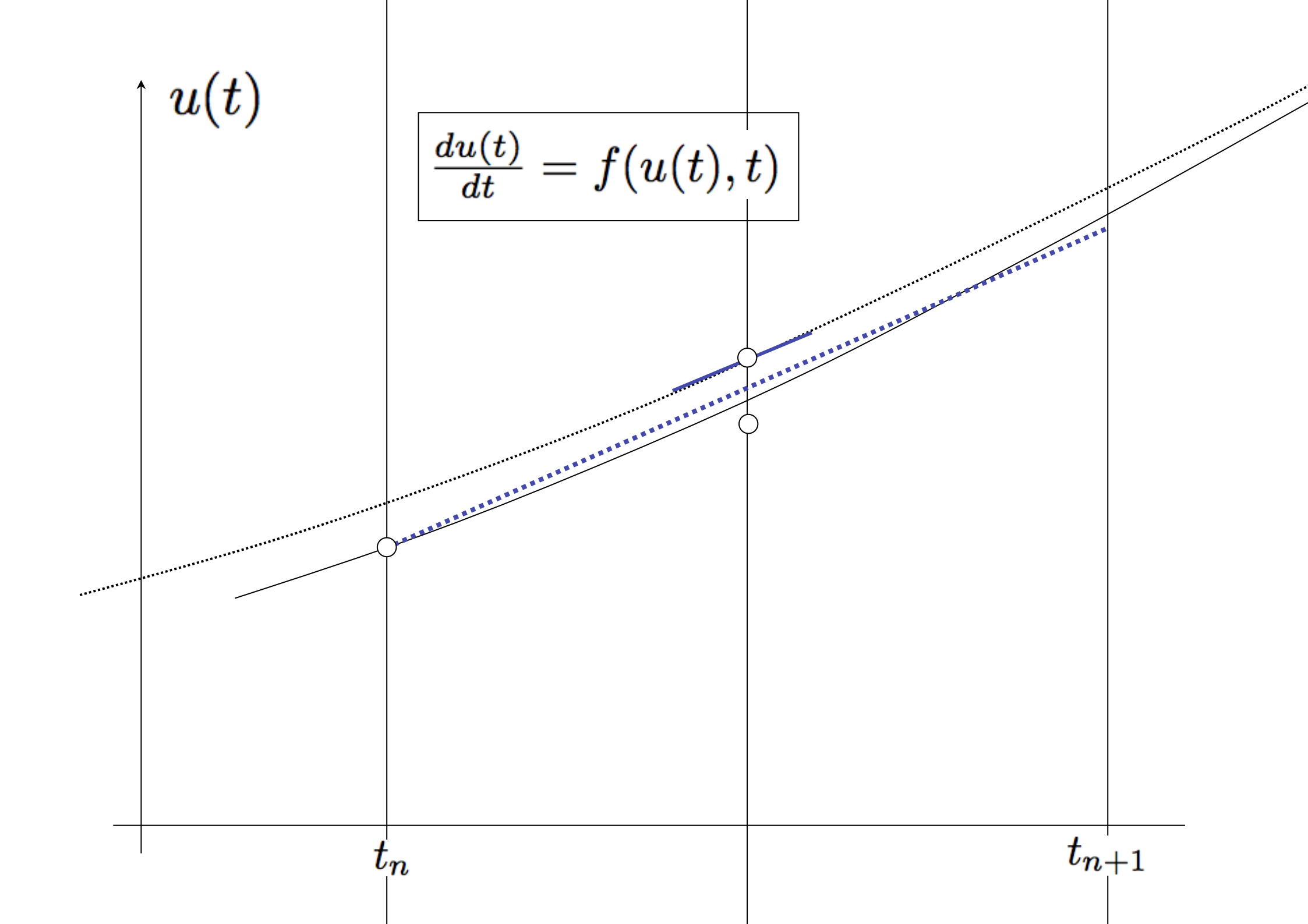

$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}

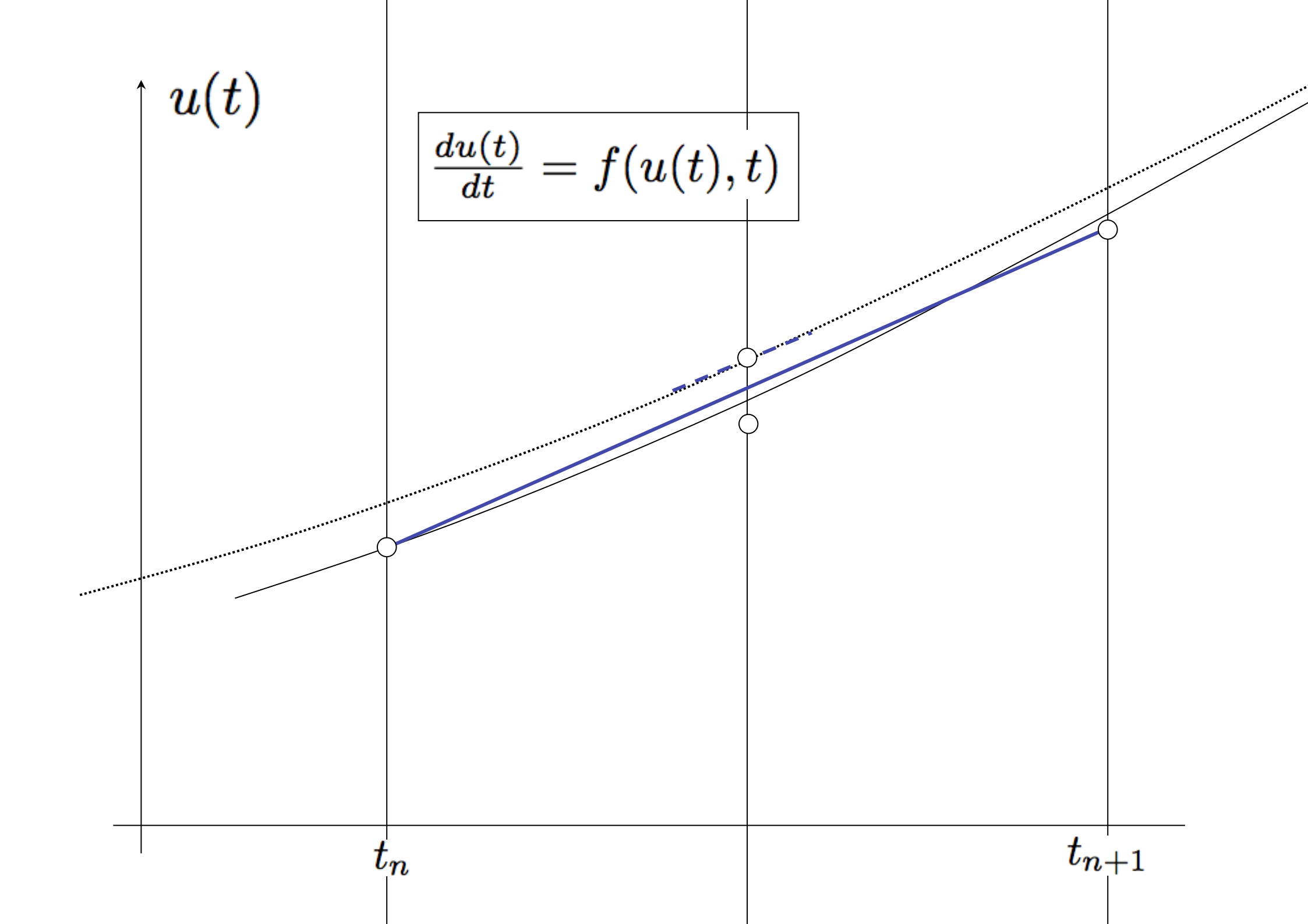

$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}

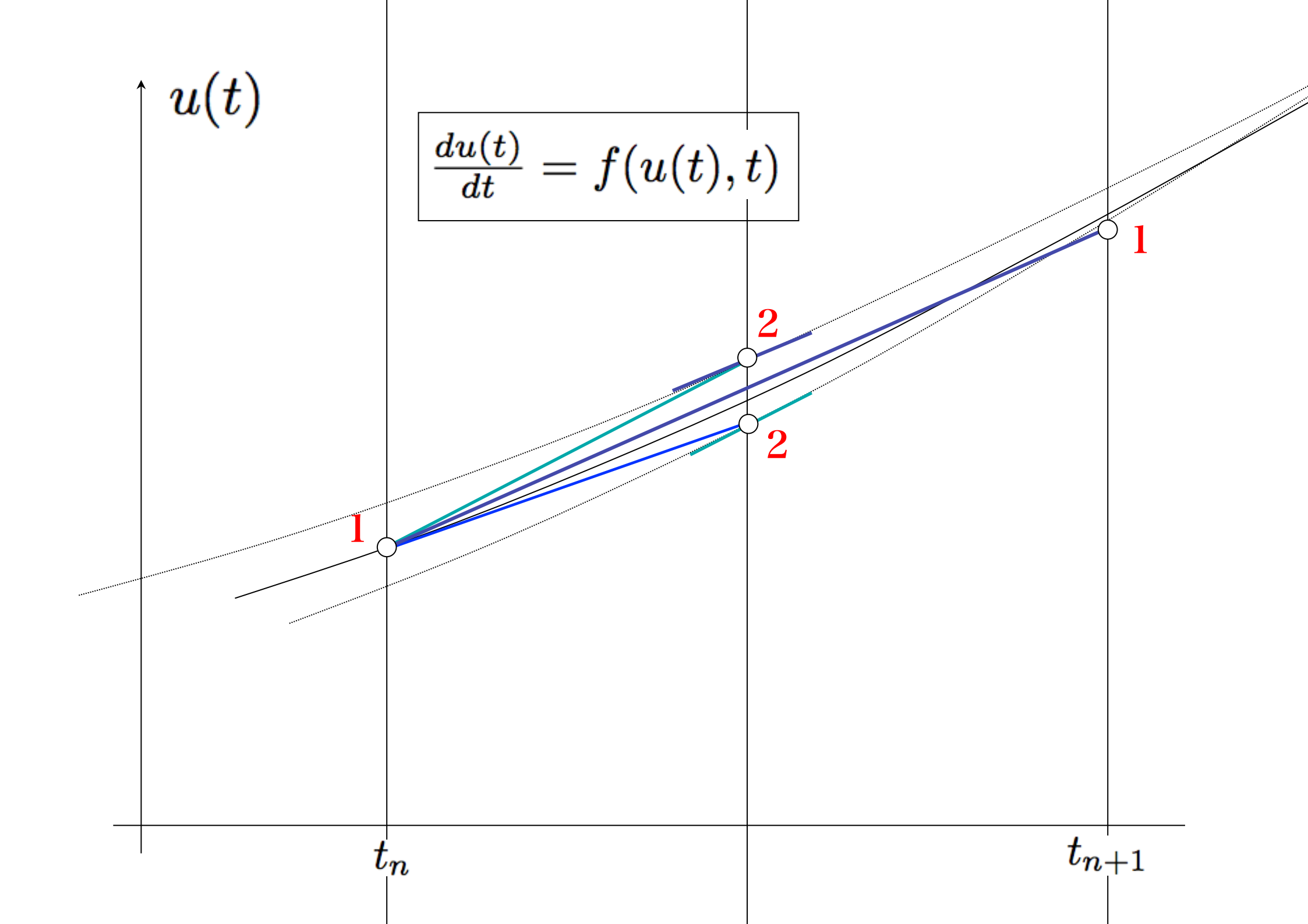

$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}


$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}


$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}


$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}


$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
t_n
t_{n+1}


$u(t)$
$\frac{du(t)}{dt} = f(u(t), t)$
1
2
2
1
t_n
t_{n+1}

4-step 4th-order Runge-Kutta Integration Method

$$\frac{du(t)}{dt} = f(t, u(t))$$

$$t_0 = t_n$$
$$u_0 = u(t_0)$$
$$df_1 = \Delta t\, f(t_0, u_0)$$

$$t_1 = t_0 + 0.5\,\Delta t$$
$$u_1 = u_0 + 0.5\, df_1$$
$$df_2 = \Delta t\, f(t_1, u_1)$$

$$t_2 = t_0 + 0.5\,\Delta t$$
$$u_2 = u_0 + 0.5\, df_2$$
$$df_3 = \Delta t\, f(t_2, u_2)$$

$$t_3 = t_0 + \Delta t$$
$$u_3 = u_0 + df_3$$
$$df_4 = f(t_3, u_3)$$

$$u_{n+1} = u_n + \tfrac{1}{6}(df_1 + 2\, df_2 + 2\, df_3 + df_4)$$

Error $O(\Delta x^5)$ for one step, $O(\Delta x^4)$ in total.

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2}$$

$u_0 = e^{ikx_j}$ ⟸ A test wave

$$df_1 = \frac{\kappa \Delta t}{(\Delta x)^2} \left\{ e^{ik(x_j+\Delta x)} - 2\, e^{ikx_j} + e^{ik(x_j-\Delta x)} \right\}$$

$$= -\frac{2\kappa \Delta t}{(\Delta x)^2} (1 - \cos k\Delta x)\, e^{ikx_j}$$

$$= -\alpha\, e^{ikx_j} \qquad \alpha = \frac{2\kappa \Delta t}{(\Delta x)^2} (1 - \cos k\Delta x)$$

$$u_1 = u_0 + 0.5\, df_1$$

$$= (1 - \tfrac{\alpha}{2})\, e^{ikx_j}$$

$$df_2 = -\frac{2\,\kappa \Delta t}{(\Delta x)^2} (1 - \tfrac{\alpha}{2}) (1 - \cos k\Delta x)\, e^{ikx_j}$$

$$= -\alpha\, (1 - \tfrac{\alpha}{2})\, e^{ikx_j}$$

$$
\begin{aligned}
u_3 &= u_0 + 0.5\, df_2 \\
&= \left(1 - \tfrac{\alpha}{2} + \tfrac{\alpha^2}{4}\right) e^{ikx_j} \\
df_3 &= -\left(\alpha - \tfrac{\alpha^2}{2} + \tfrac{\alpha^3}{4}\right) e^{ikx_j} \\
u_4 &= u_0 + df_3 \\
&= \left(1 - \alpha + \tfrac{\alpha^2}{2} - \tfrac{\alpha^3}{4}\right) e^{ikx_j} \\
df_4 &= -\left(\alpha - \alpha^2 + \tfrac{\alpha^3}{2} - \tfrac{\alpha^4}{4}\right) e^{ikx_j} \\
u_{\text{new}} &= u_0 + \tfrac{1}{6}\left(df_1 + 2\, df_2 + 2\, df_3 + df_4\right) \\
&= \left(1 - \alpha + \tfrac{\alpha^2}{2} - \tfrac{\alpha^3}{6} + \tfrac{\alpha^4}{24}\right) e^{ikx_j} \\
&= \left\{1 + \tfrac{(-\alpha)}{1!} + \tfrac{(-\alpha)^2}{2!} + \tfrac{(-\alpha)^3}{3!} + \tfrac{(-\alpha)^4}{4!}\right\} e^{ikx_j}
\end{aligned}
$$

By one step integration of 4-th order Runge-Kutta method,

$$u_{\text{new}} = \left\{1 + \frac{(-\alpha)}{1!} + \frac{(-\alpha)^2}{2!} + \frac{(-\alpha)^3}{3!} + \frac{(-\alpha)^4}{4!}\right\} e^{ikx_j}$$

$$\alpha = \frac{2\kappa\Delta t}{(\Delta x)^2}(1 - \cos k\Delta x)$$

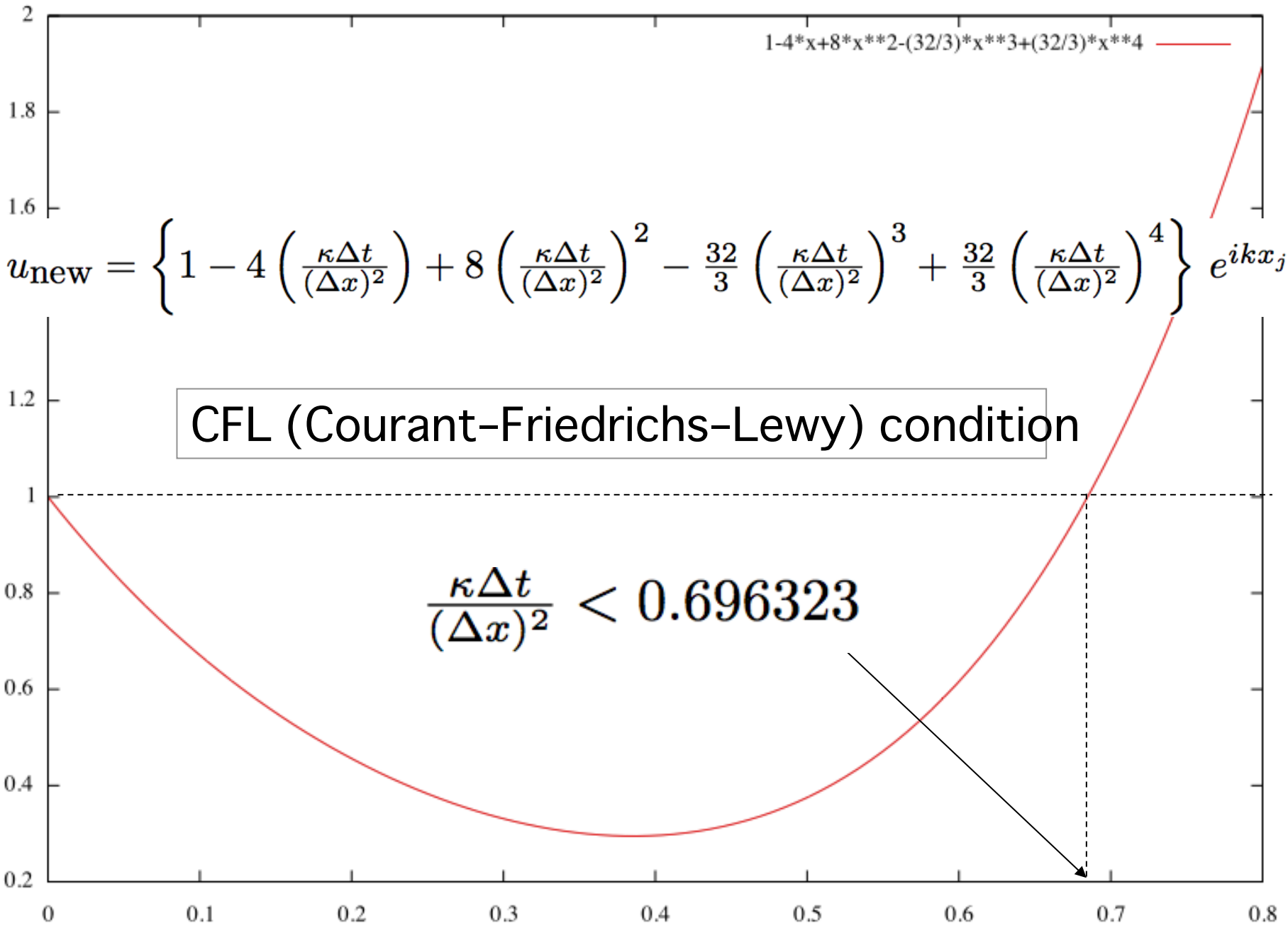
When $k\Delta x << 1$ $\quad \alpha \sim \frac{\kappa\Delta t}{(\Delta x)^2}(k\Delta x)^2 = k^2\,\kappa\Delta t$

$$u_{\text{exact}} = e^{-k^2\,\kappa\Delta t}\, e^{ikx_j} = e^{-\alpha\,\Delta t}$$

Error in 1step $= O[(\Delta t)^5]$ $\Longrightarrow$ Error in total $= O[(\Delta t)^4]$

When $k\Delta x = \pi$, $\quad \alpha = \frac{4\kappa\Delta t}{(\Delta x)^2}$

$$u_{\text{new}} = \left\{1 - 4\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right) + 8\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^2 - \frac{32}{3}\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^3 + \frac{32}{3}\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^4\right\} e^{ikx_j}$$



$$u_{\text{new}} = \left\{ 1 - 4\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right) + 8\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^2 - \frac{32}{3}\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^3 + \frac{32}{3}\left(\frac{\kappa\Delta t}{(\Delta x)^2}\right)^4 \right\} e^{ikx_j}$$

CFL (Courant–Friedrichs–Lewy) condition

$$\frac{\kappa\Delta t}{(\Delta x)^2} < 0.696323$$

Simple Numerical Simulation with Fortran90 Code

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

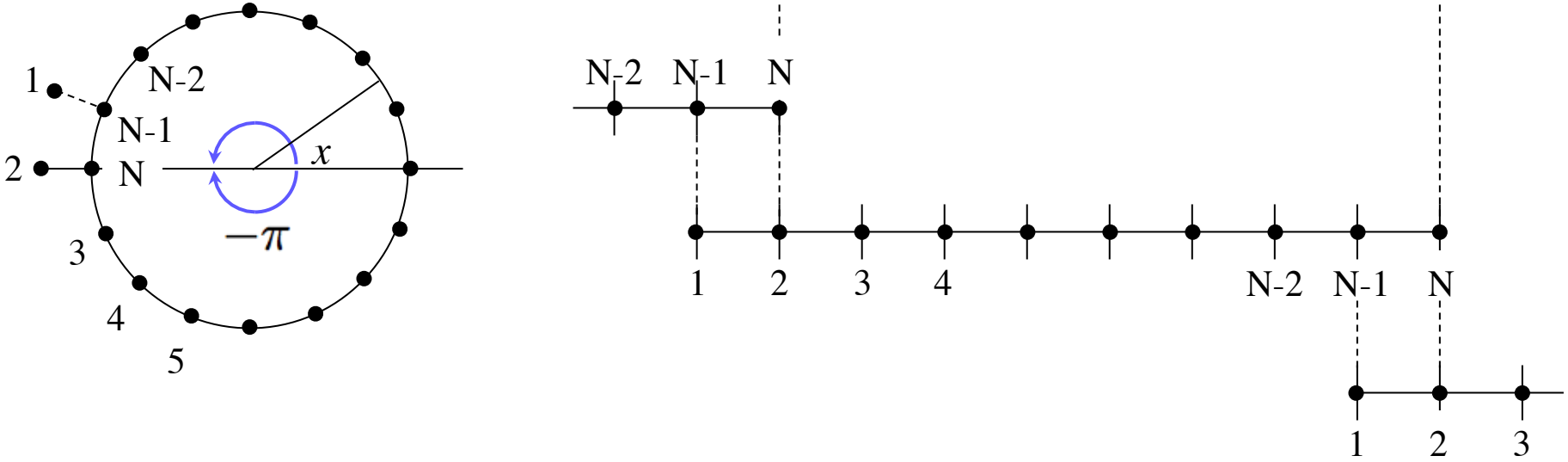
A Sample Code in FDM

$$\frac{d\psi_j}{dt} = \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

$$\frac{d\psi_j}{dt} = f(\psi_1, \psi_2, \cdots, \psi_N)$$

==> 4-step (4th order) Runge-Kutta method

Periodic boundary condition




```
subroutine iBoundary_condition(psi)
   real(DP), dimension(nx), intent(inout) :: psi

   psi(1)    = psi(nx-1)
   psi(nx)   = psi(2)

 end subroutine iBoundary_condition
```

Now let's see the code: main.f90

```
do nloop = 1 , nloop_max

   dpsi01(:) = rk4__step('1st',dt,dx,psi)
   call iBoundary_condition(dpsi01)

   dpsi02(:) = rk4__step('2nd',dt,dx,psi,dpsi01)
   call iBoundary_condition(dpsi02)

   dpsi03(:) = rk4__step('3rd',dt,dx,psi,dpsi02)
   call iBoundary_condition(dpsi03)

   dpsi04(:) = rk4__step('4th',dt,dx,psi,dpsi03)
   call iBoundary_condition(dpsi04)

    time = time + dt
    psi(:) = psi(:) + ONE_SIXTH*(dpsi01(:)       &
                                +2*dpsi02(:)     &
                                +2*dpsi03(:)     &
                                  +dpsi04(:))
end do
```

Runge-Kutta step (rk.f90)

```
function rk4__step(nth,dt,dx,psi,dpsi_prev)       &
                                    result(dpsi_new)

    character(len=3), intent(in)          :: nth
    real(DP), intent(in)                  :: dt
    real(DP), intent(in)                  :: dx
    real(DP), dimension(:), intent(in)    :: psi
    real(DP), dimension(size(psi,dim=1)),    &
                    intent(in), optional  :: dpsi_prev
    real(DP), dimension(size(psi,dim=1)) :: dpsi_new
    real(DP), dimension(size(psi,dim=1)) :: psi_
```

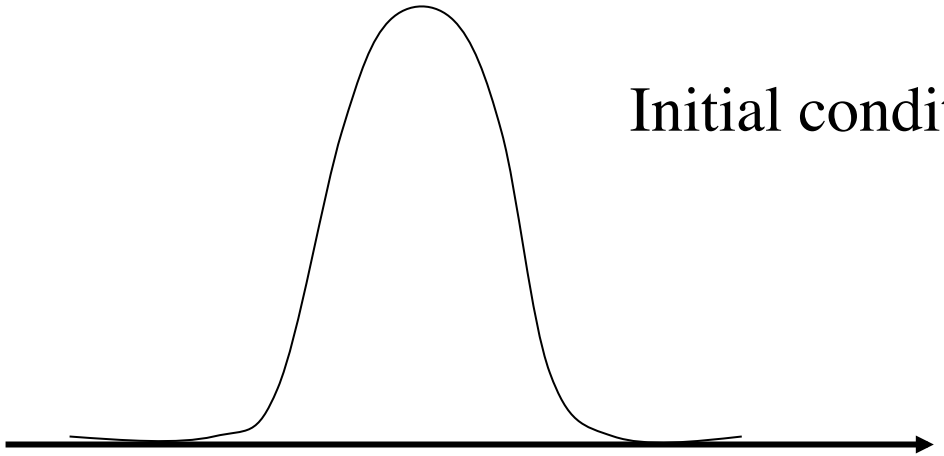
```
  select case (nth)
  case ('1st')
    dpsi_new(:) = dt*diffusion_equation(size(psi,dim=1), &
                                        dx,psi)
  case ('2nd')
    psi_(:) = psi(:) + dpsi_prev(:)*0.5_DP
    dpsi_new(:) = dt*diffusion_equation(size(psi,dim=1), &
                                        dx,psi_)
  case ('3rd')
    psi_(:) = psi(:) + dpsi_prev(:)*0.5_DP
    dpsi_new(:) = dt*diffusion_equation(size(psi,dim=1), &
                                        dx,psi_)
  case ('4th')
    psi_(:) = psi(:) + dpsi_prev(:)
    dpsi_new(:) = dt*diffusion_equation(size(psi,dim=1), &
                                        dx,psi_)
  end select

end function rk4__step
```

diffusion_equation (rk.f90)

```
function diffusion_equation(nx,dx,psi)
   integer, intent(in)                  :: nx
   real(DP), intent(in)                 :: dx
   real(DP), dimension(nx), intent(in) :: psi
   real(DP), dimension(nx)              :: diffusion_equation

   integer :: i
   real(DP) :: dx2

   dx2 = namelist__double('Diffusion_coeff')/(dx**2)

   do i = 2 , nx-1
      diffusion_equation(i) = dx2*(psi(i+1)-2*psi(i)+psi(i-1))
   end do

 end function diffusion_equation
```

$$\kappa \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

Let's run the code



src/DiffusionEquation

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial x^2}$$

Time Integration in Fourier Spectral Method

$$\boxed{\frac{d\psi_m(t)}{dt} = -m^2\,\psi_m(t)}$$

$$\psi_m(t) = \psi_m(0)\,\exp(-m^2\,t)$$

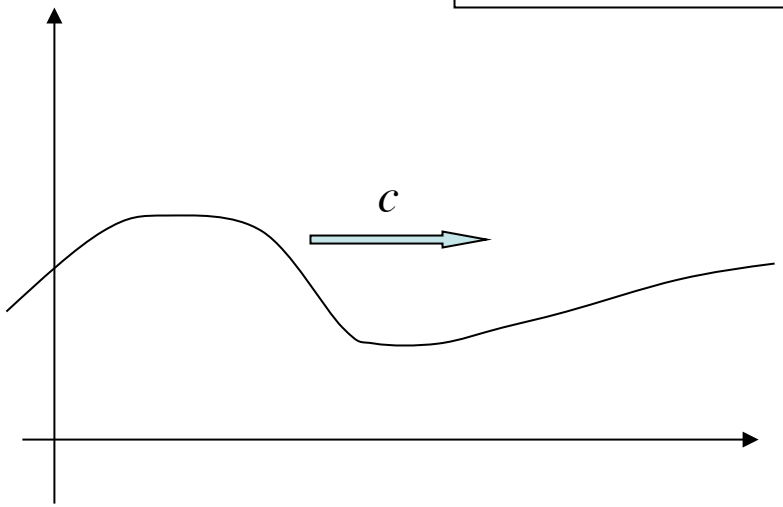
$$\Longrightarrow \quad \psi(x,t) = \sum_m \psi_m(0)\,\frac{\exp(imx - m^2\,t)}{\sqrt{2\pi}}$$

This is an exceptionally lucky case, it's not always that simple, especially…

Nonlinear terms in spectral methods

Burgers' equation

$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x} + \nu \frac{\partial^2 \psi}{\partial x^2}$$



$$\frac{\partial \psi}{\partial t} = -c \frac{\partial \psi}{\partial x}$$

Solution:

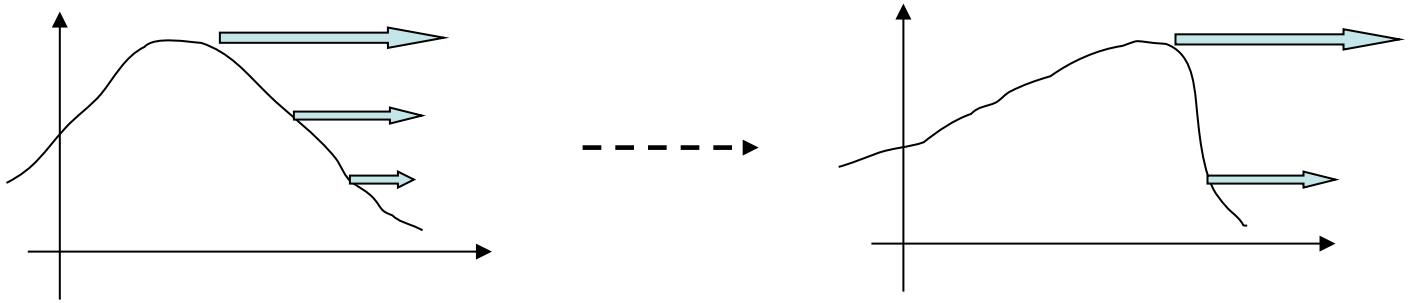
$$\psi(x,t) = f(x - ct)$$

Burgers' equation

$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x} + \nu \frac{\partial^2 \psi}{\partial x^2}$$

Diffusion term

$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x}$$



To solve Burgers' equation by spectral methods

$$\psi(x,t) = \sum_k \psi_k(t) \frac{\exp(ikx)}{\sqrt{2\pi}}$$

$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x} + \nu \frac{\partial^2 \psi}{\partial x^2}$$

$$\Rightarrow \quad \frac{d\psi_m(t)}{dt} = -\sum_k \frac{\psi_k(t)\psi_{m-k}(t)}{\sqrt{2\pi}} - \nu\, m^2\, \psi_m(t)$$

convolution

To solve Burgers' equation by Spectral Methods

M set of equations

$$\frac{d\psi_m(t)}{dt} = -\underbrace{\sum_k \frac{\psi_k(t)\psi_{m-k}(t)}{\sqrt{2\pi}}}_{M \text{ multiplications}} - \nu\, m^2\, \psi_m(t)$$

$$\vdots$$

$$O(M^2) \text{ calcuations}$$

Pseudo-spectral Method

$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x} + \nu \frac{\partial^2 \psi}{\partial x^2}$$

Fourier space	Real space
$\psi_m(t), \quad im\,\psi_m(t)$	$\psi(x,t), \quad \frac{\partial \psi}{\partial x}(x,t)$
	$g(x,t) \equiv \psi(x,t)\frac{\partial \psi}{\partial x}(x,t)$
$g_m(t)$	

Fourier Transform

$$\frac{d\psi_m}{dt} = -g_m(t) - m^2\,\psi_m(t)$$

Fast Fourier Transform (FFT)

Fourier space	Real space
f_m	$f_j(x) = \sum_m f_m \frac{\exp(imx)}{\sqrt{2\pi}}$
$f_m = \int_{-\pi}^{\pi} f_j(x) \frac{\exp(-imx)}{\sqrt{2\pi}}\, dx$	$f_j(x)$

$$O(M \log M) \quad \text{calculations}$$

$$<< O(M^2)$$

Nonlinear terms: FDM

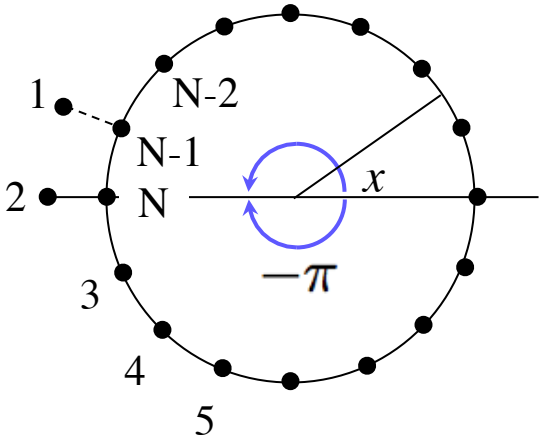
$$\frac{\partial \psi}{\partial t} = -\psi \frac{\partial \psi}{\partial x} + \nu \frac{\partial^2 \psi}{\partial x^2}$$

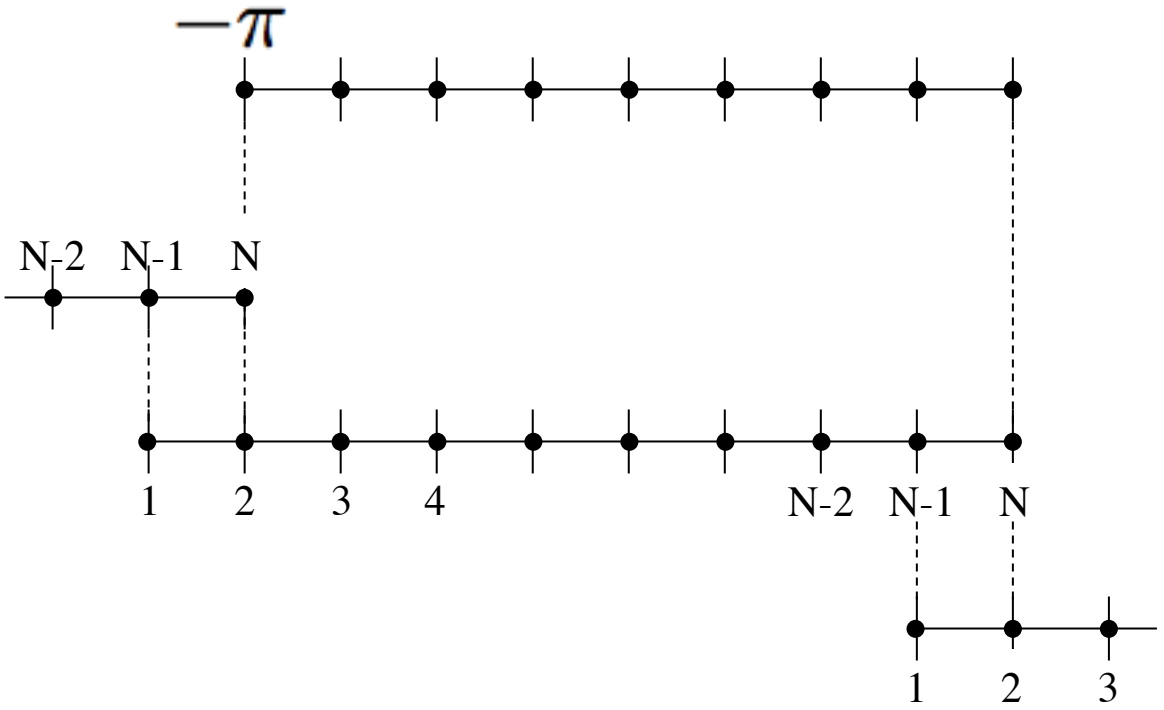
$$\Rightarrow \quad \frac{d\psi_j}{dt} = -\psi_j \frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x} + \nu \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

Let's see the code.

code
- src/BurgersEquation

Burgers' equation by FDM





$$\frac{d\psi_j}{dt} = -\psi_j \frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x} + \nu \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

Runge-Kutta 1st step (rk4.f90)

```
function rk4__1st(dt,dx,psi) result(dpsi01)
 real(DP), intent(in)                  :: dt
 real(DP), intent(in)                  :: dx
 real(DP), dimension(:), intent(in)    :: psi
 real(DP), dimension(size(psi,dim=1)) :: dpsi01

 real(DP), dimension(size(psi,dim=1)) :: psi_

 dpsi01(:) = dt*burgers_equation(size(psi,dim=1),dx,psi)

end function rk4__1st
```

burgers_equation (rk4.f90)

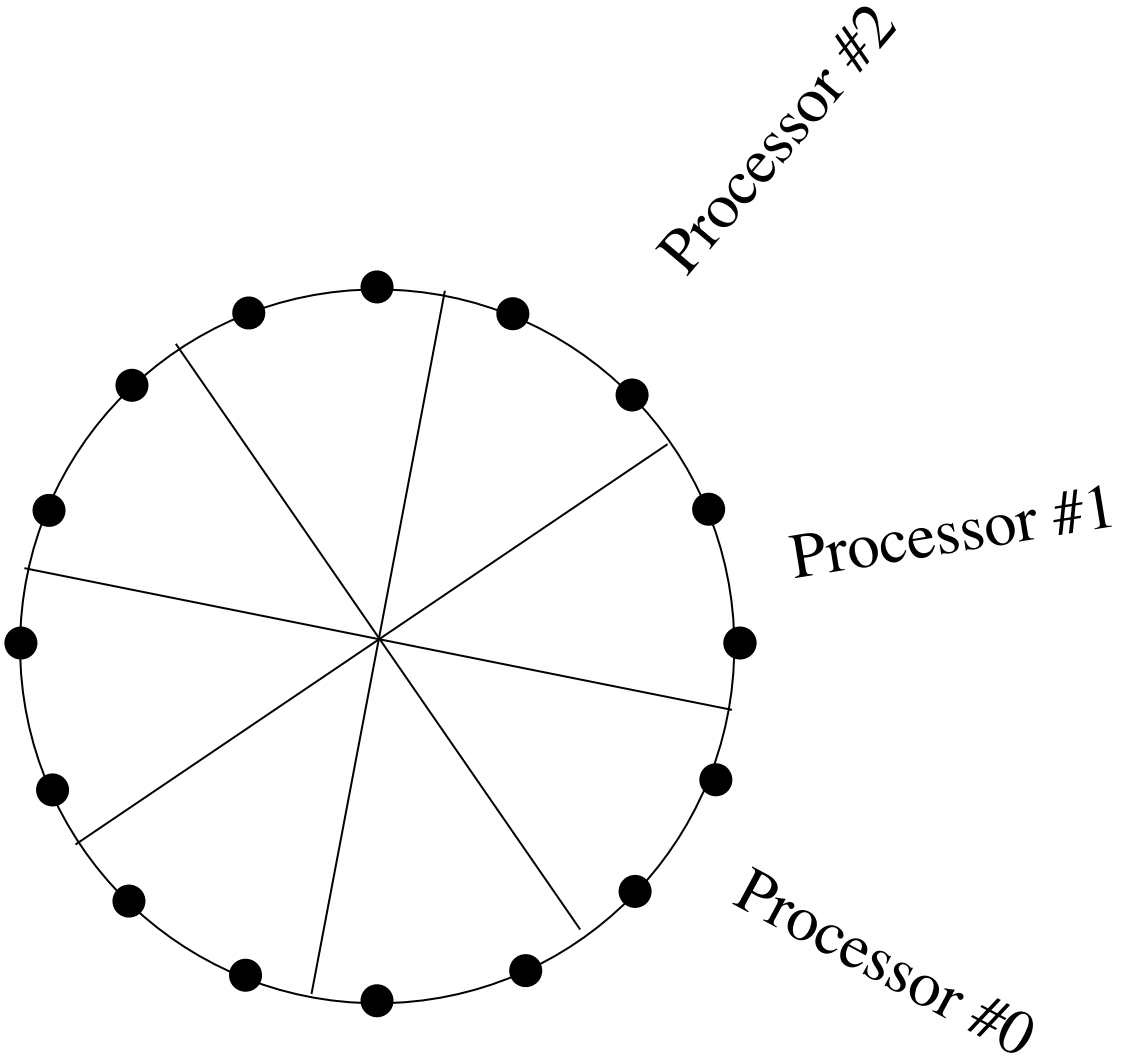
```
function burgers_equation(nx,dx,psi)
  integer, intent(in)                    :: nx
  real(DP), intent(in)                   :: dx
  real(DP), dimension(nx), intent(in) :: psi
  real(DP), dimension(nx)                :: burgers_equation

  integer :: i
  real(DP) :: dx1, dx2

  dx1 = 1.0_DP / (2*dx)
  dx2 = namelist__double('Diffusion_coeff')/(dx**2)

  do i = 2 , nx-1
    burgers_equation(i) = - psi(i)*dx1*(psi(i+1)-psi(i-1)) &
                     + dx2*(psi(i+1)-2*psi(i)+psi(i-1))
  end do

end function burgers_equation
```

$$\frac{d\psi_j}{dt} = -\psi_j \frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x} + \nu \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2}$$

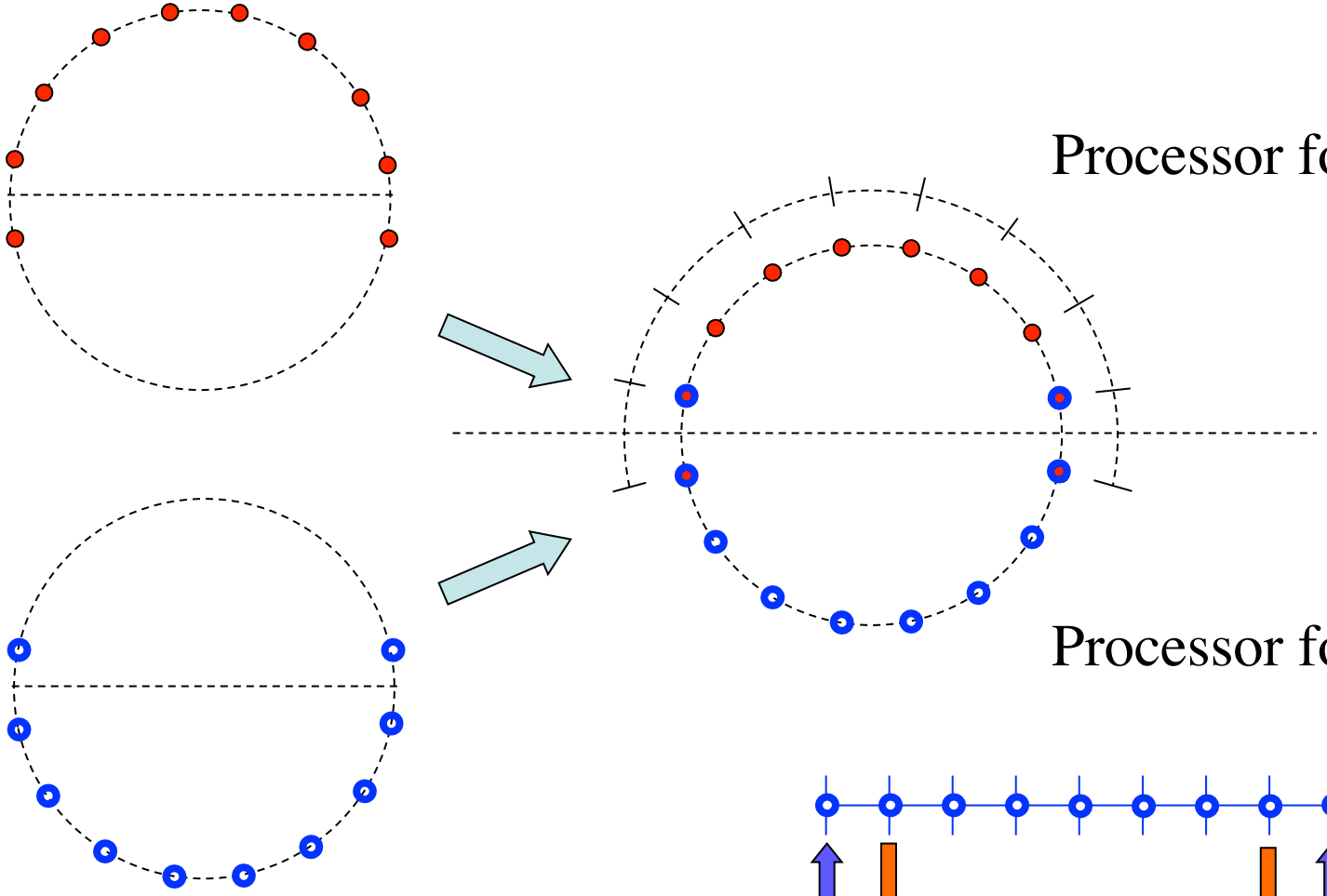
Let’s run the code

src/BuergersEquation

Parallel processing by domain decomposition



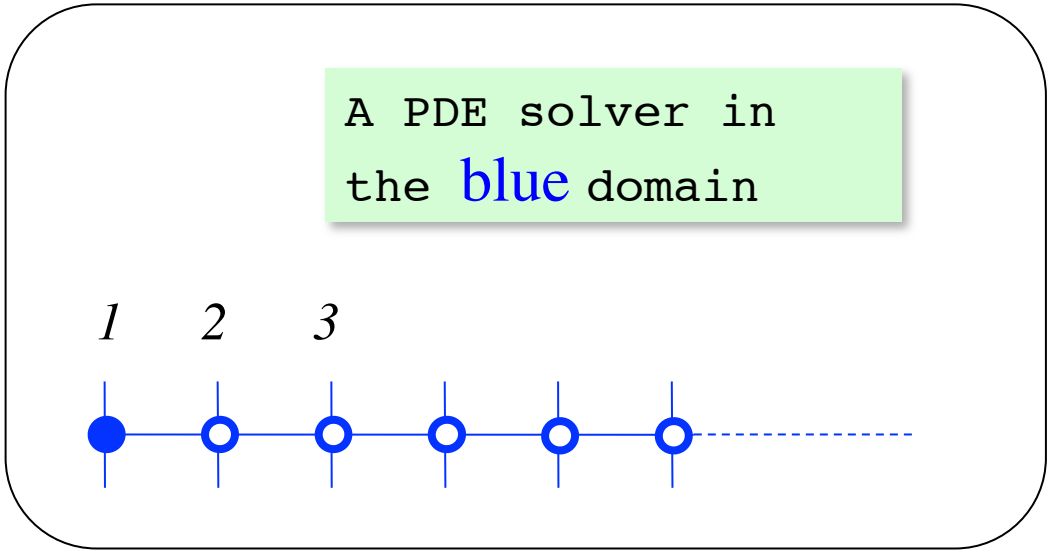
Parallel computation by 2 processors

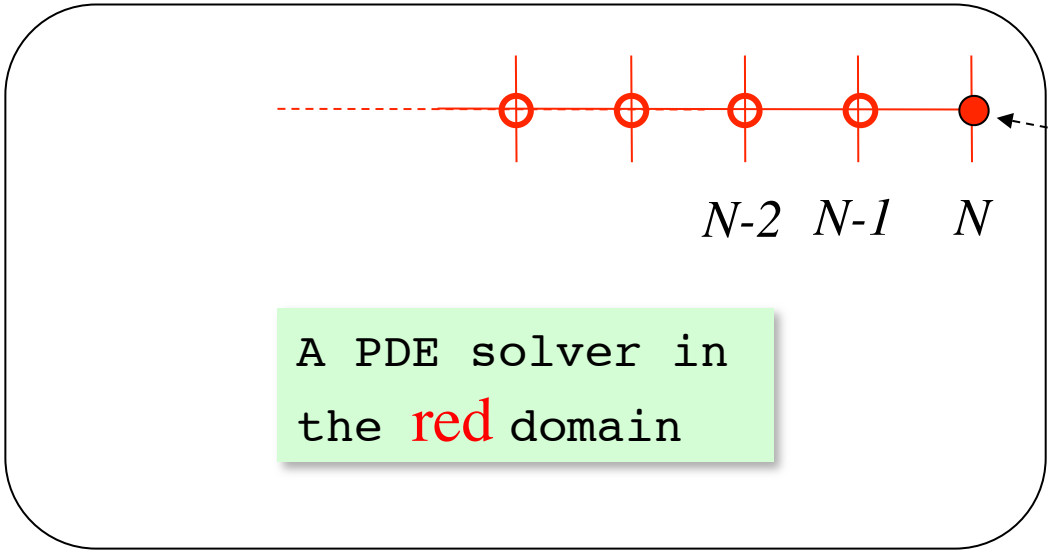


Processor for red domain.

Processor for blue domain.

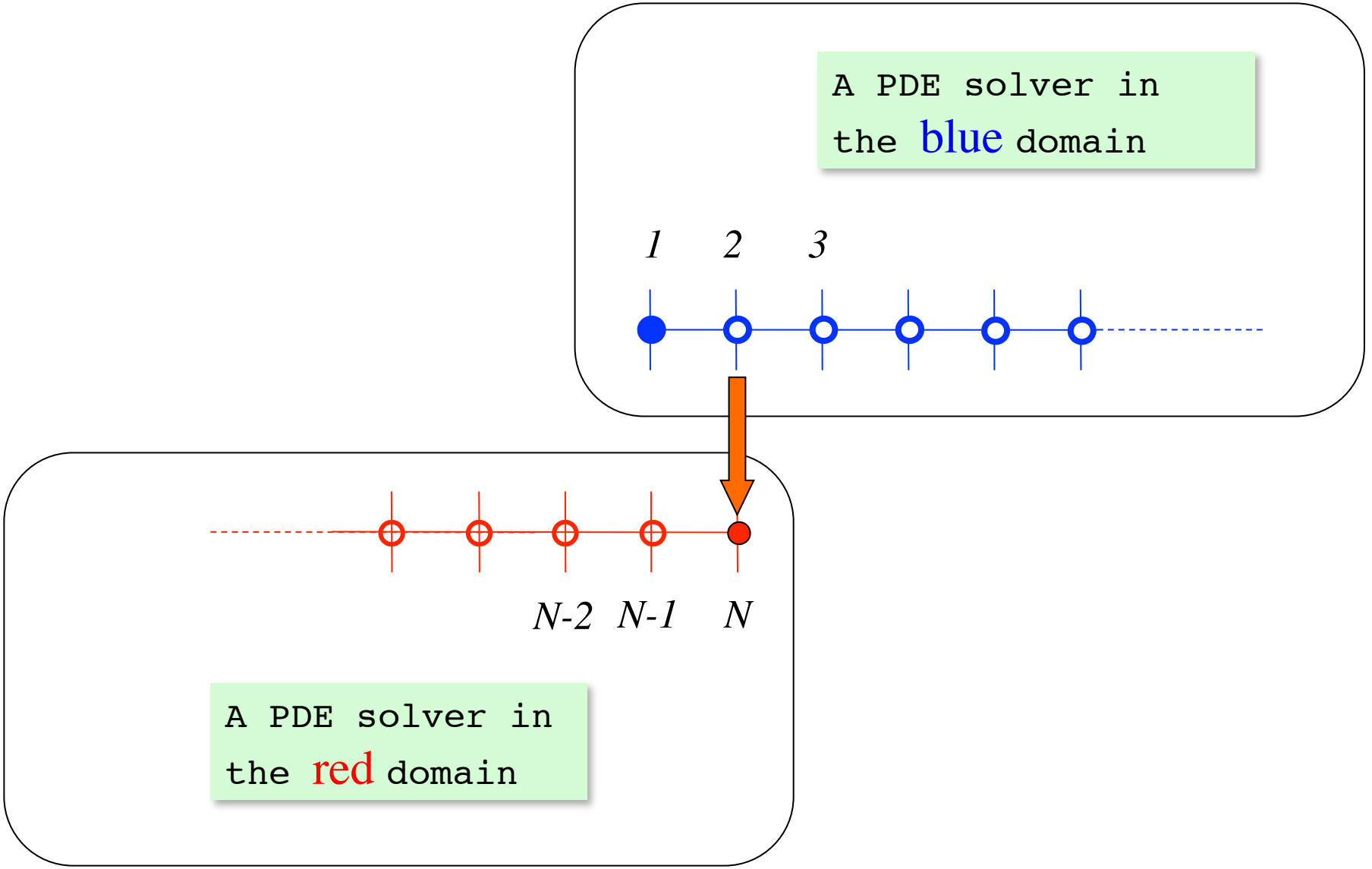
Inter process communication



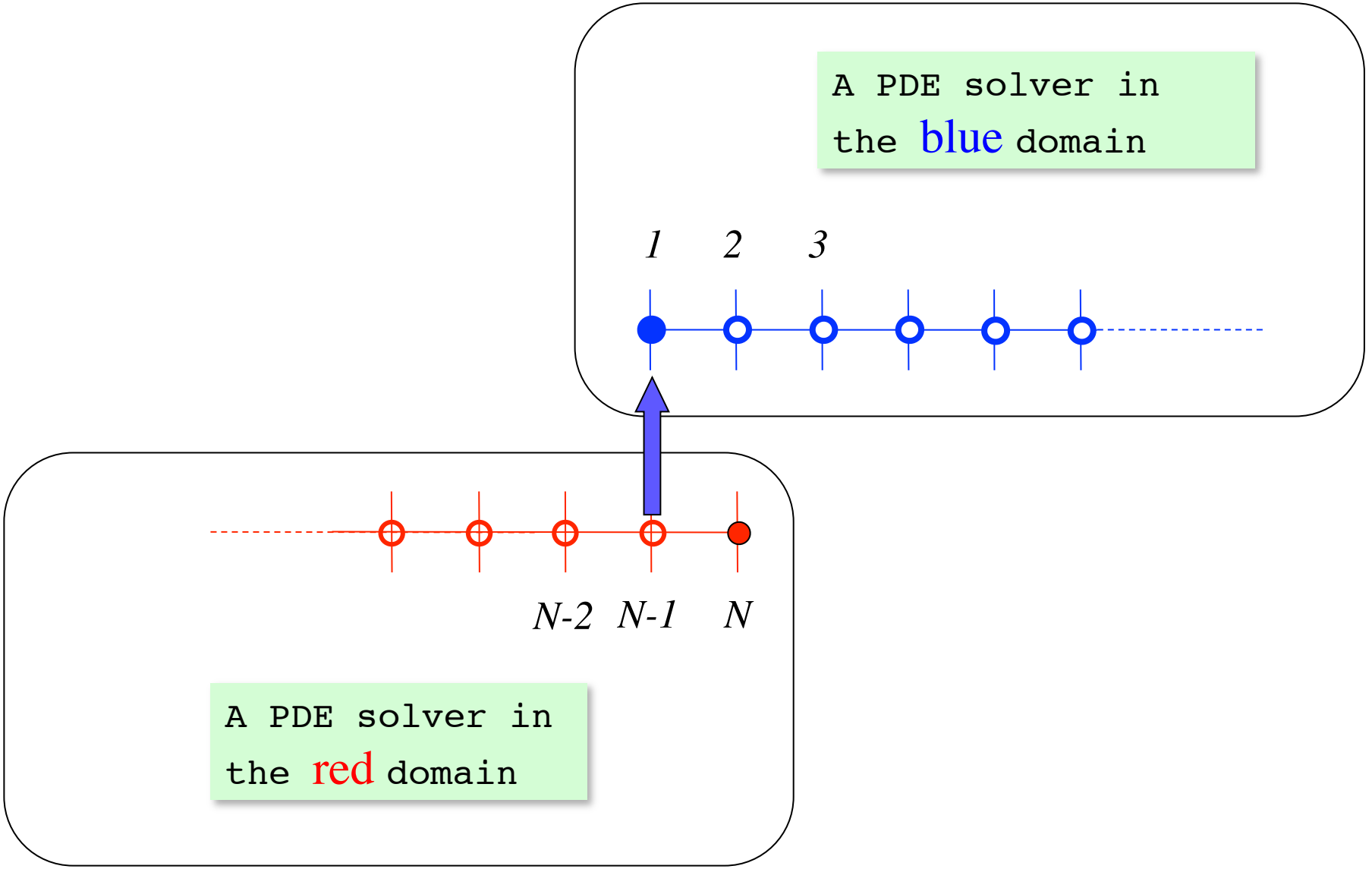


Set the boundary value
from other solver.

Inter process communication



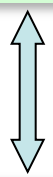
Inter process communication



In the code

In the red processor:

```
do i = 2 , nx-1
  diffusion_equation(i) = dx2*(psi(i+1)-2*psi(i)+psi(i-1))
end do
```



Same computation, same code.

In the blue processor:

```
do i = 2 , nx-1
  diffusion_equation(i) = dx2*(psi(i+1)-2*psi(i)+psi(i-1))
end do
```

In the code

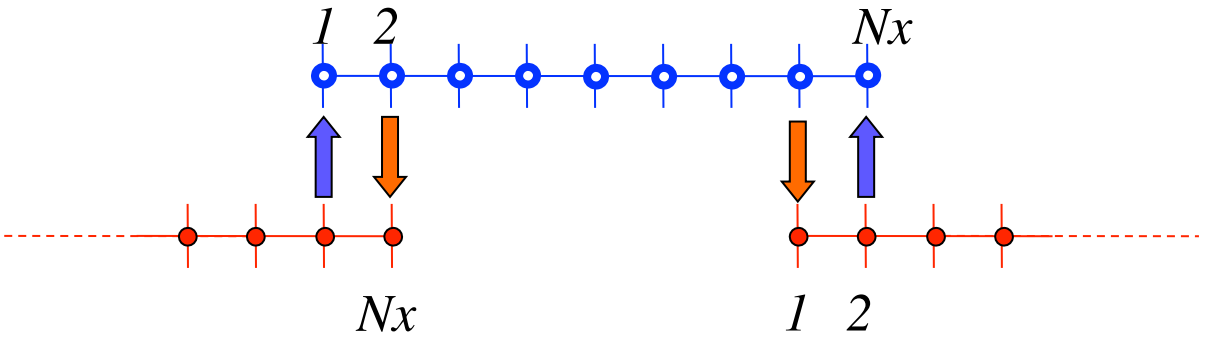
In the red processor:

```
do i = 2 , nx-1
  diffusion_equation(i) = dx2*(psi(i+1)-2*psi(i)+psi(i-1))
end do
```

Same computation, same code.

In the blue processor:

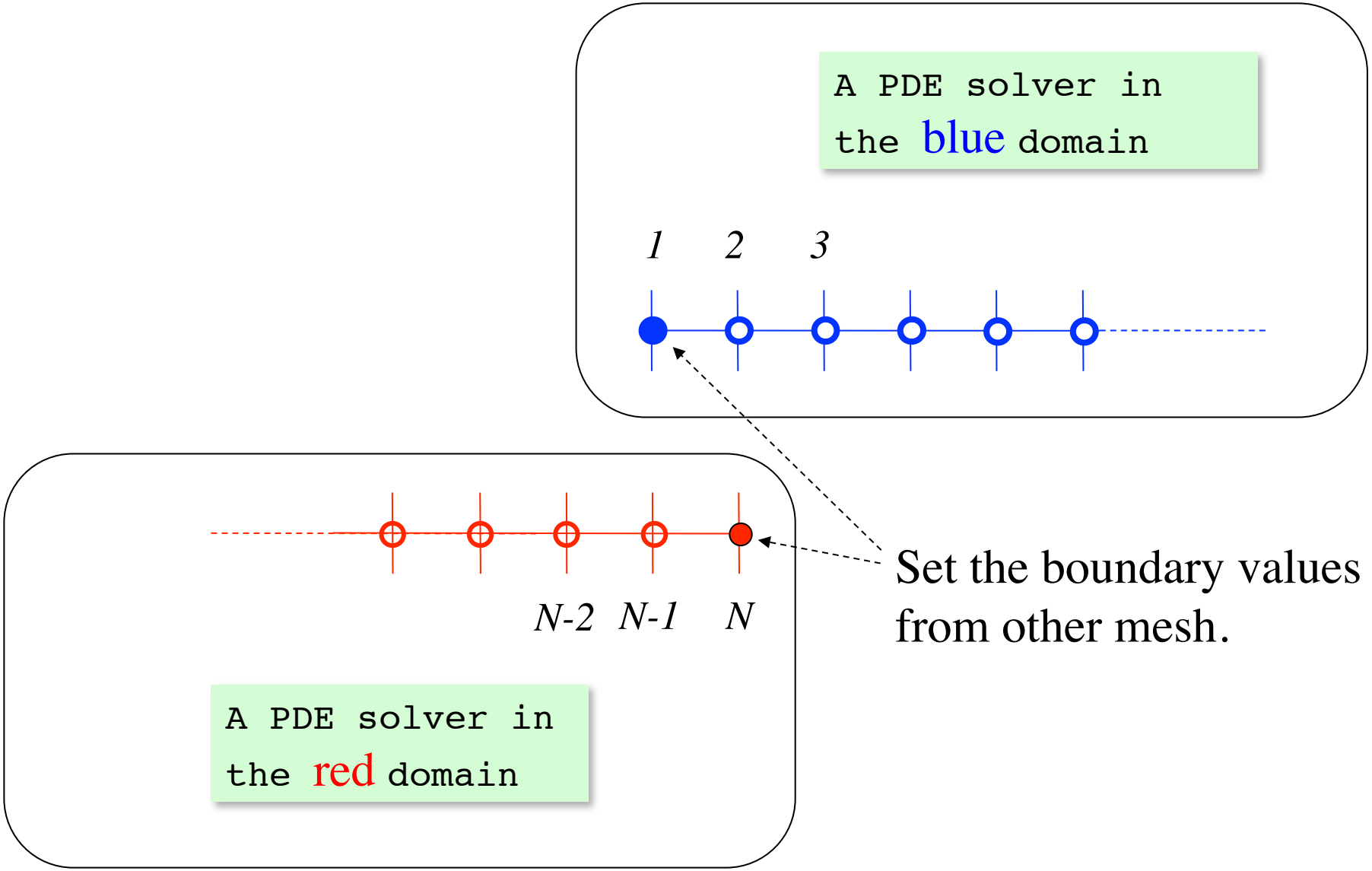
```
do i = 2 , nx-1
  diffusion_equation(i) = dx2*(psi(i+1)-2*psi(i)+psi(i-1))
end do
```



"Boundary condition"

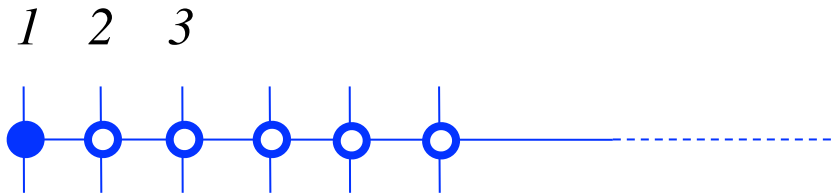
Inter-process communication (by MPI, etc.)

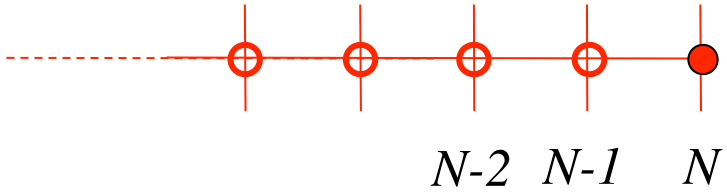
Two independent PDE solvers



Two independent PDE solvers: Grid size can be different

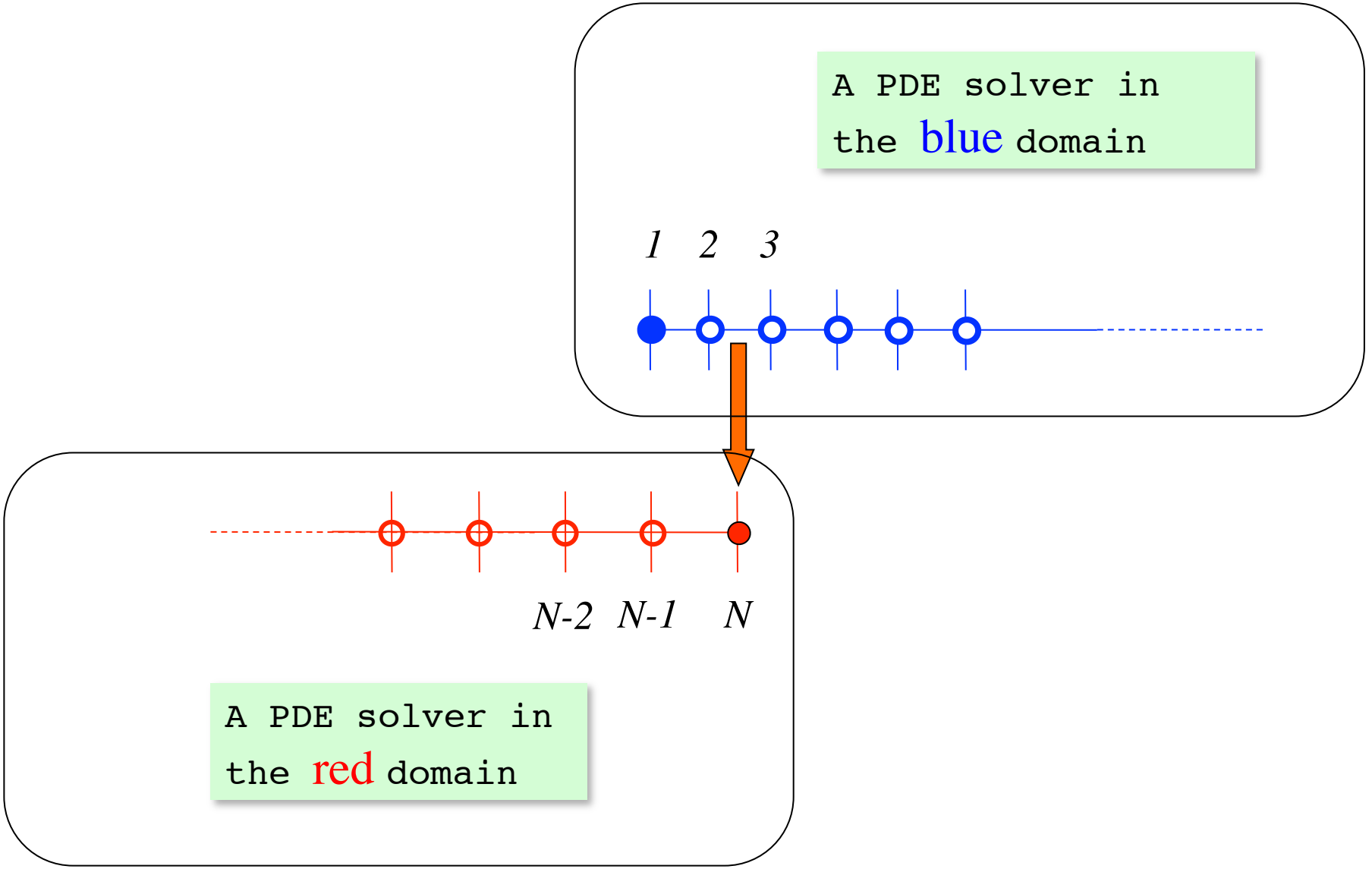
A PDE solver in the blue domain



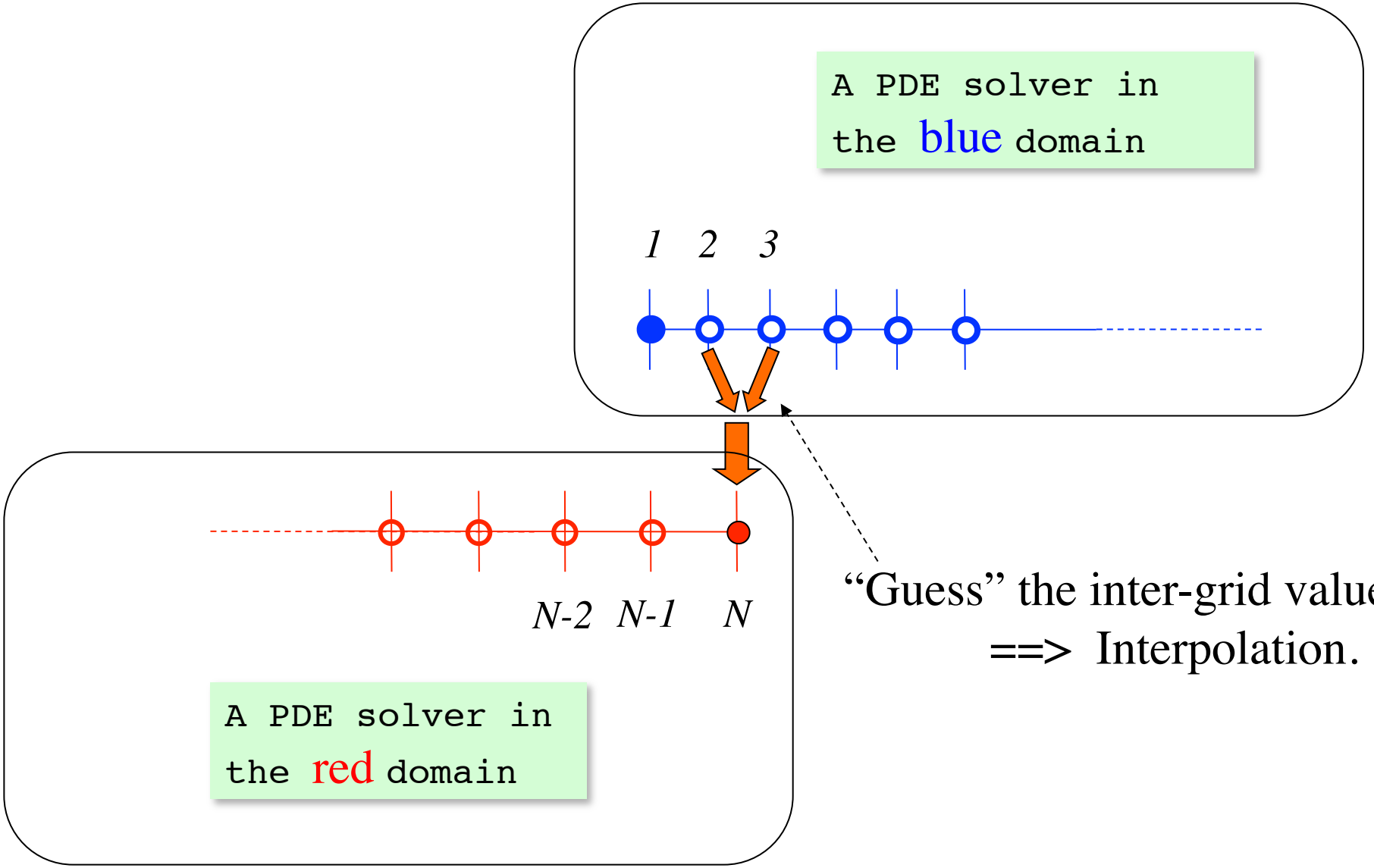


A PDE solver in the red domain

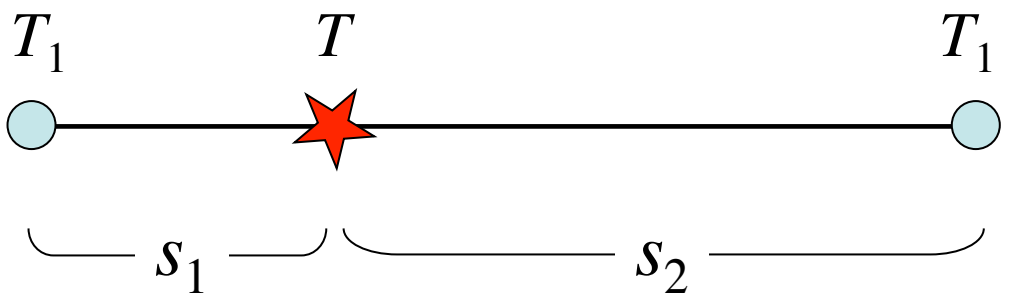
Two independent PDE solvers: Grid size can be different



No problem

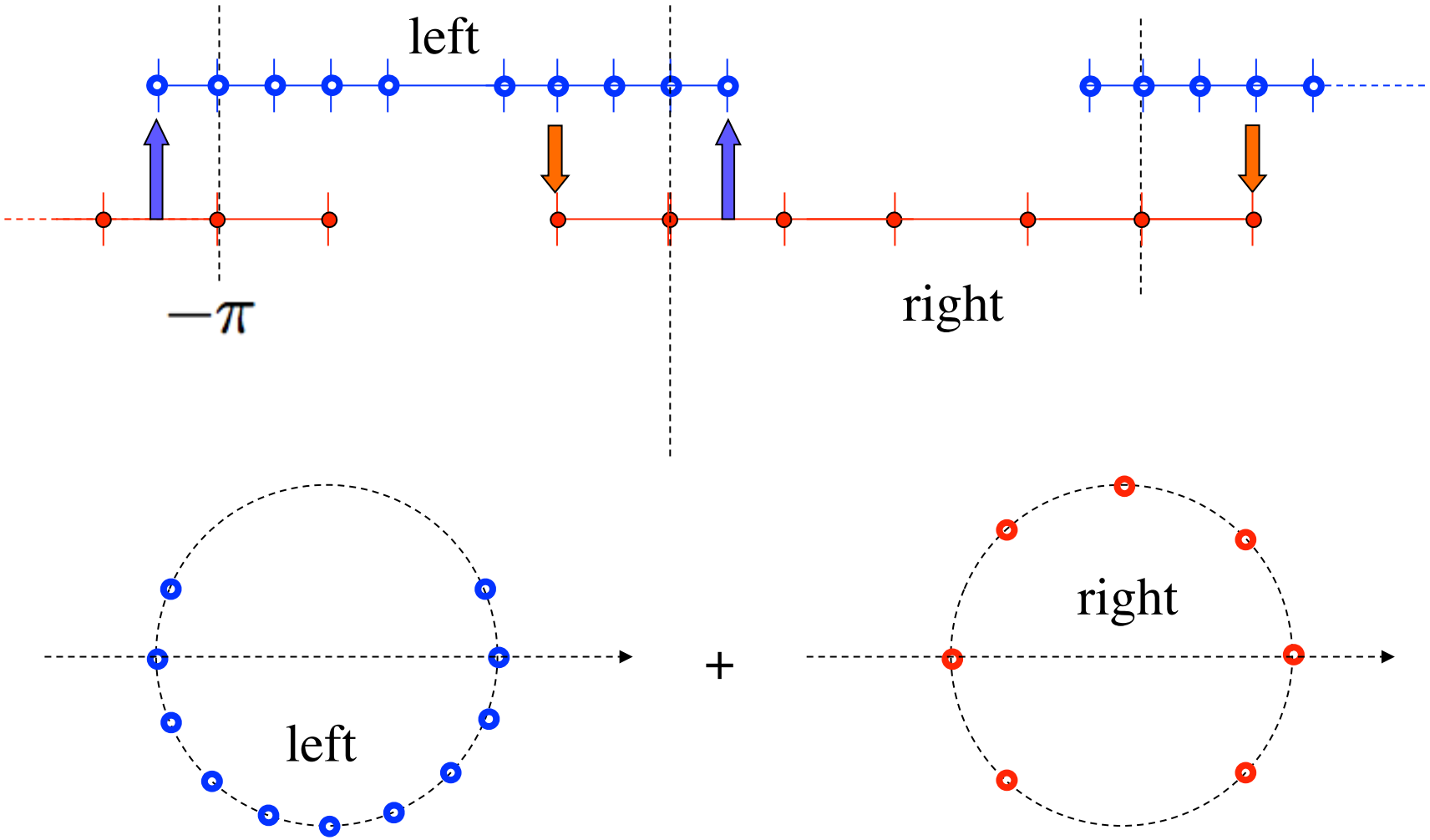

A PDE solver in the blue domain
1 2 3
N-2 N-1 N
A PDE solver in the red domain
"Guess" the inter-grid value.
==> Interpolation.

Linear interpolation in 1-D



$$T = (s_2 T_1 + s_1 T_2) / (s_1 + s_2)$$

Overset grid method : 1-D Burgers' equation



Let’s run the code

Code

- src/SampleChimera

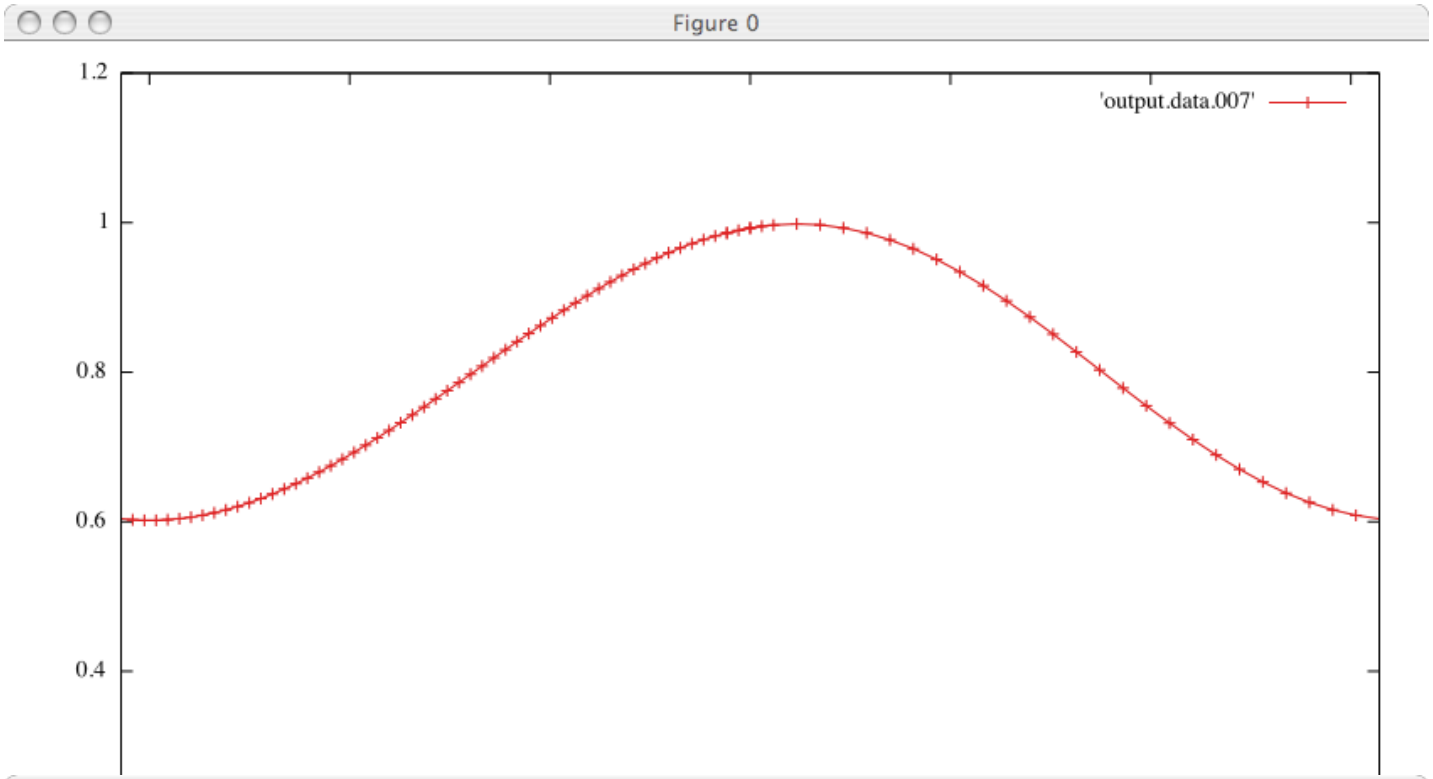


Figure 0
1.2
1
0.8
0.6
0.4
'output.data.007'

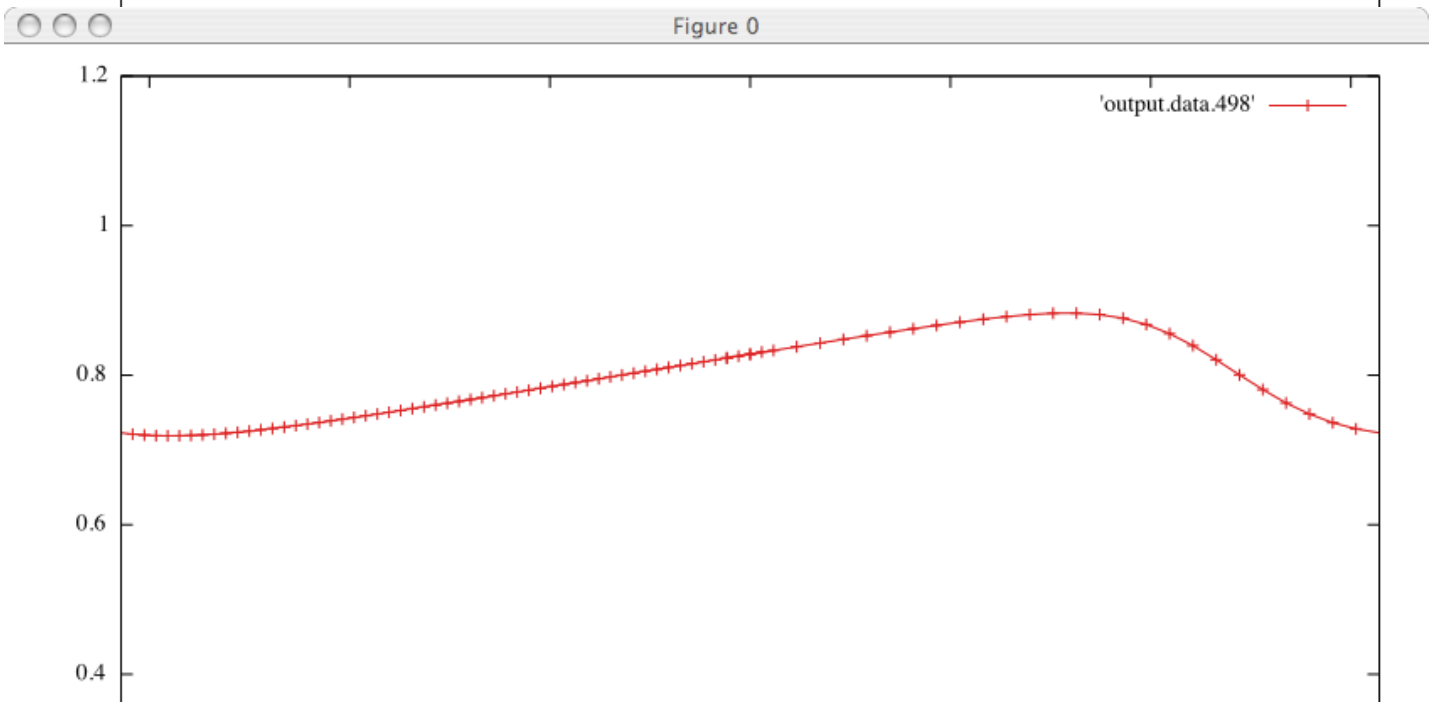
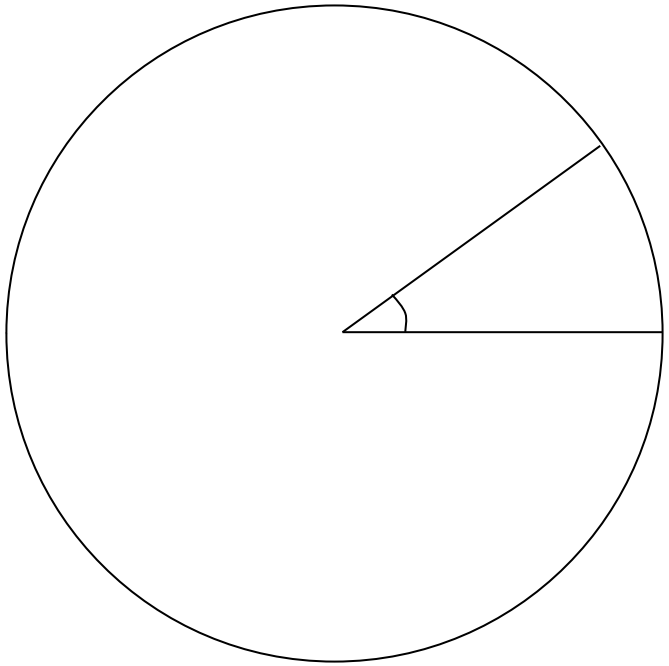


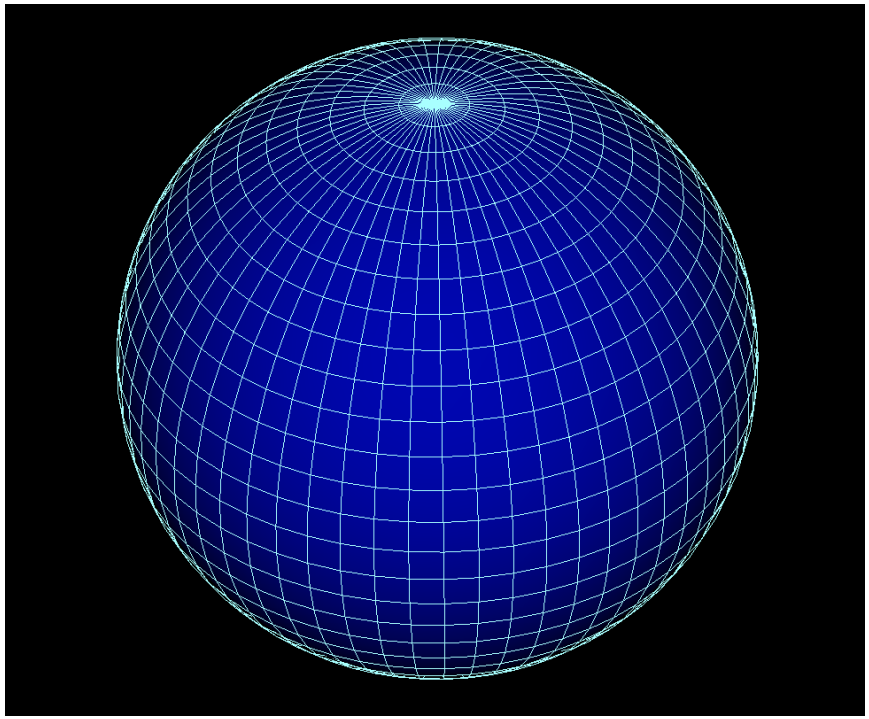
Figure 0
1.2
1
0.8
0.6
0.4
'output.data.498'

Numerical Methods on a Sphere

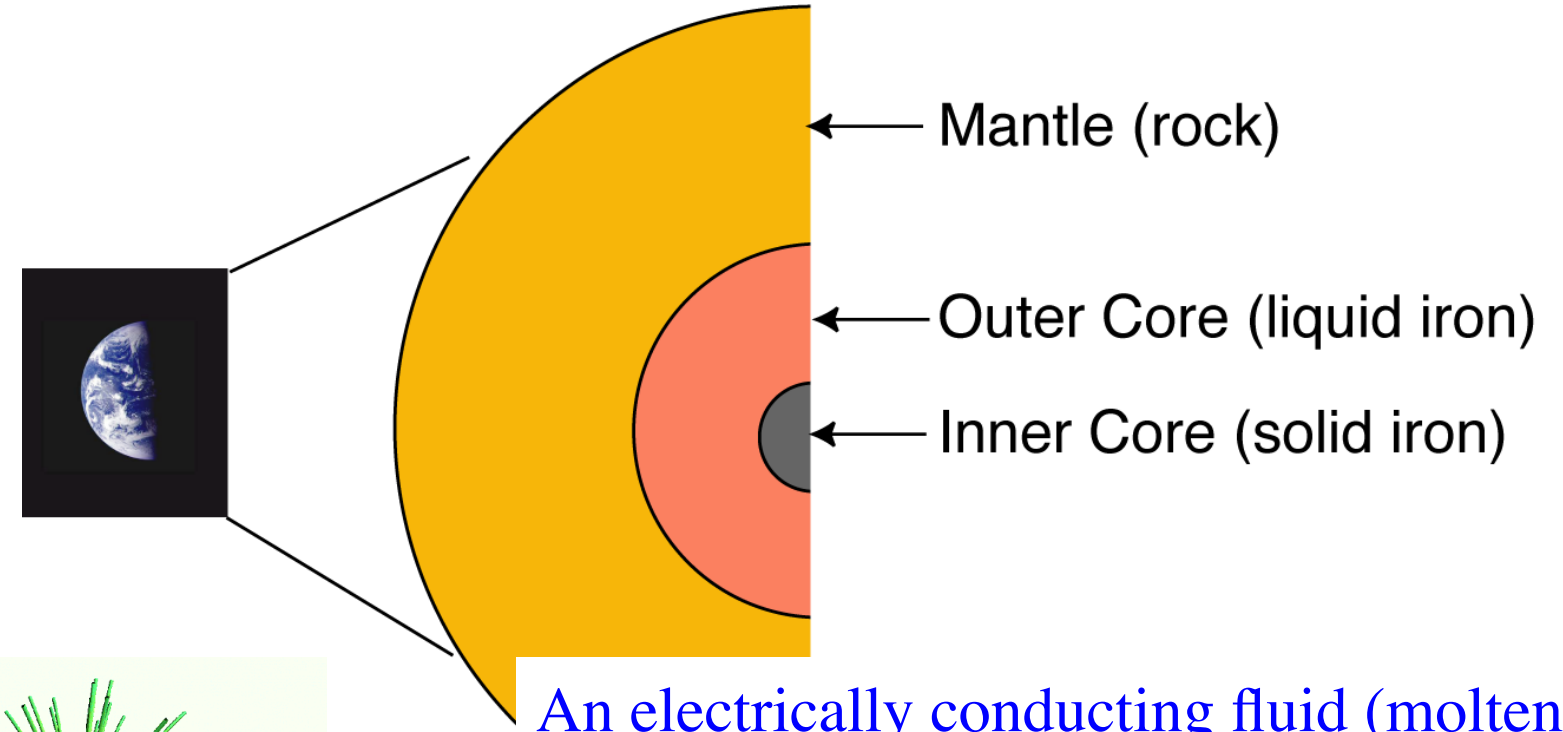
2D… circle

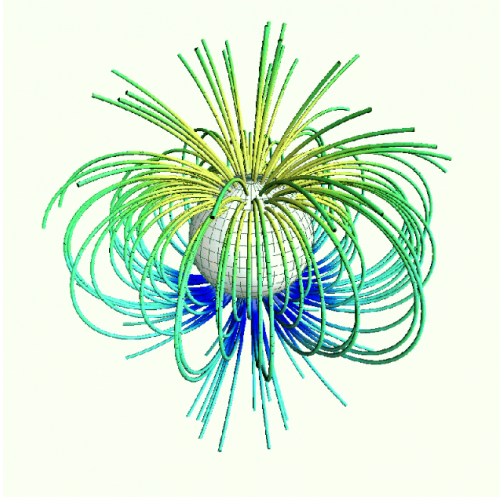


3D… sphere



MHD Fluid in the Earth





An electrically conducting fluid (molten iron).

- MHD eqs.
- Convection
- Magnetic field (electric current) generation
- MHD dynamo

Coordinate singularity causes two different problems

1. On the poles:

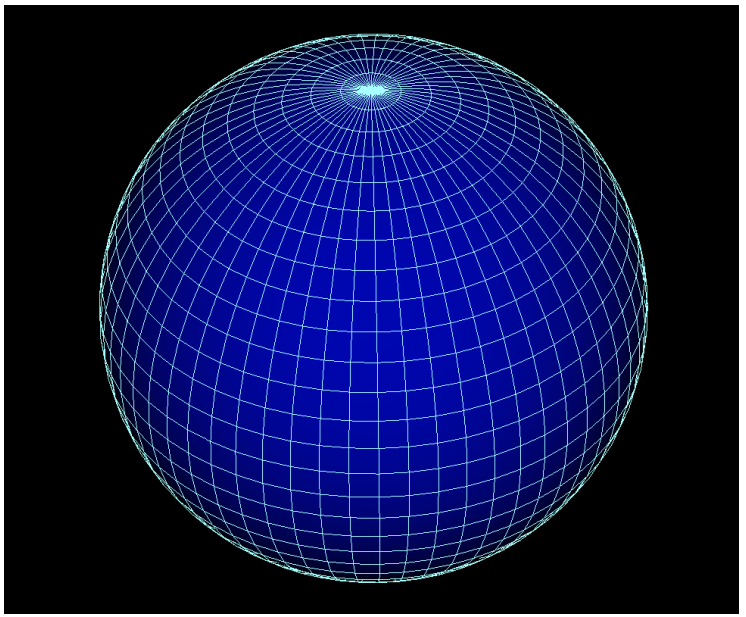
   - Need's special cares;
     e.g., L'Hospital's theorem.

   - Not serious.

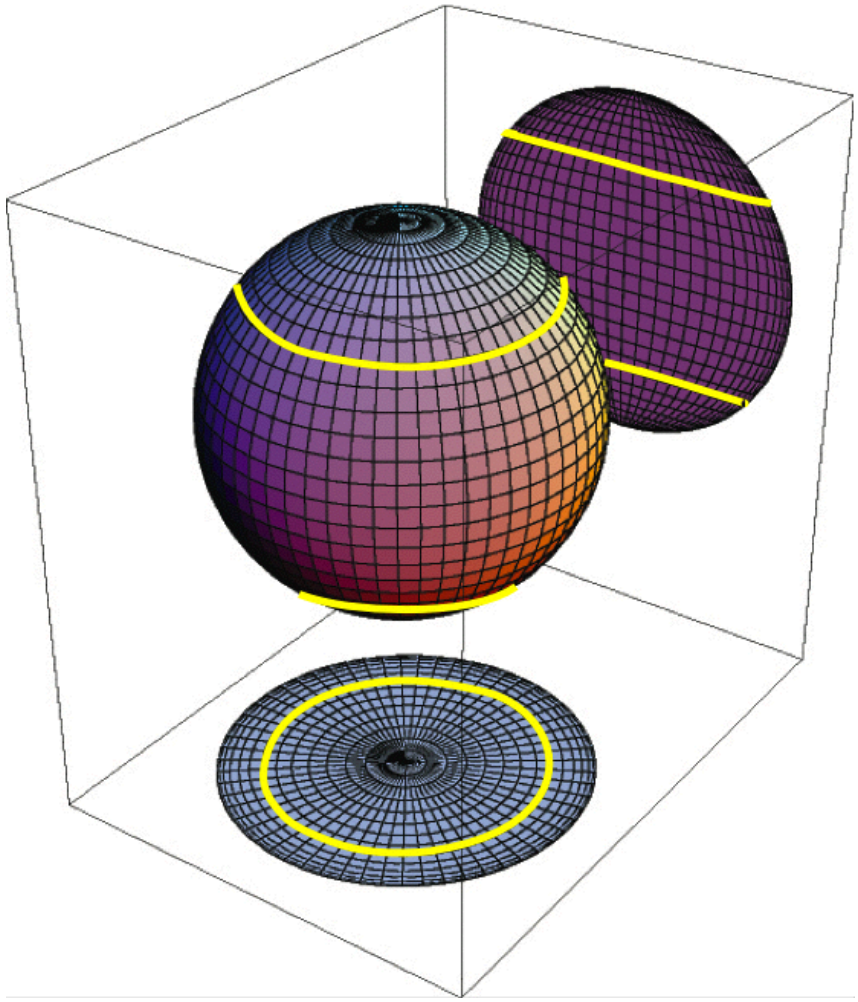
2. Near the poles:

   - Grid convergence.

   - Serious; waste of CPU time.



Grid convergence = inefficiency

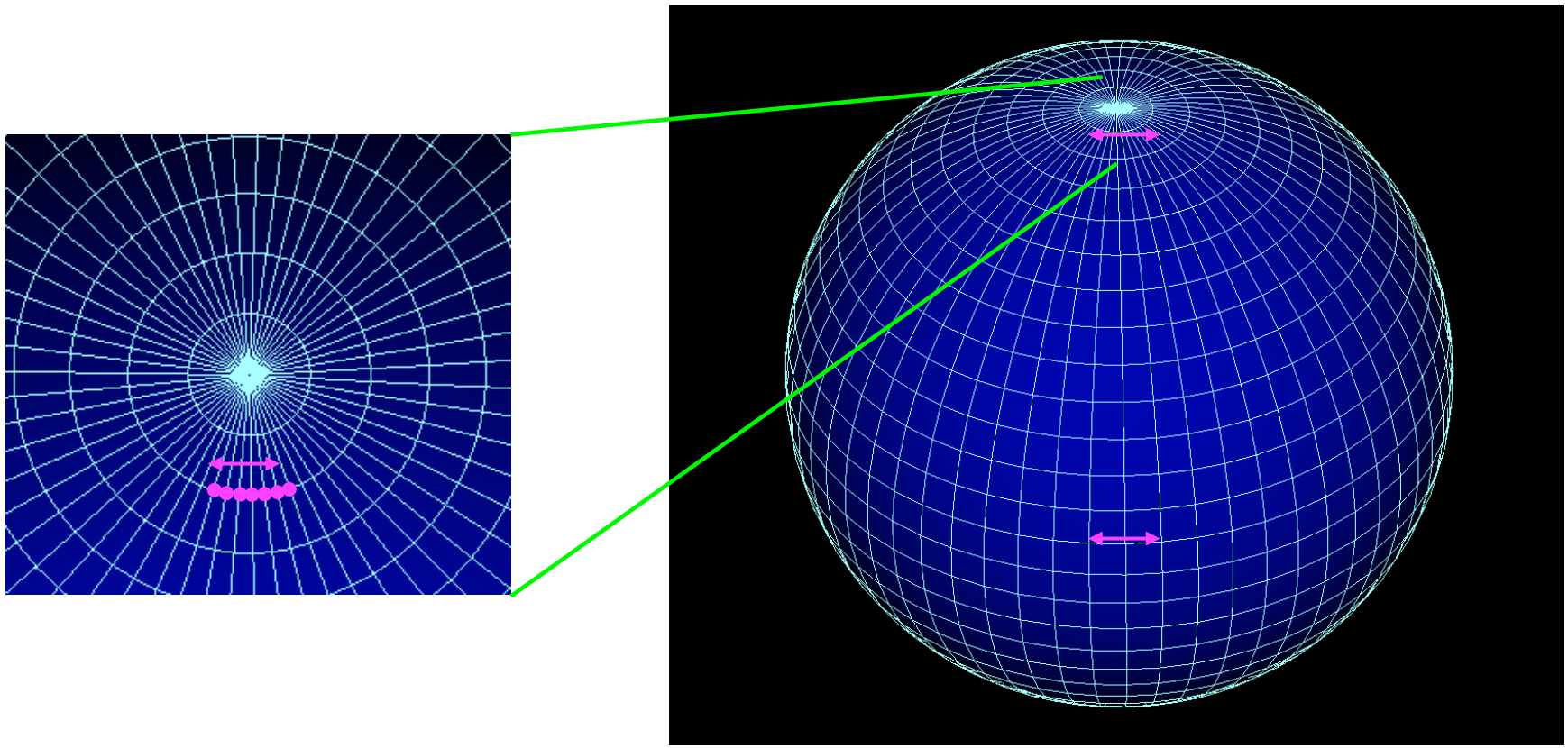


84% of grid points are located in high-latitude part (>45^{o} N and S).

Low latitude part (between 45^{o} N and S) is covered by only 16%.

Grid Spacing Problem in Lat-Lon Grid

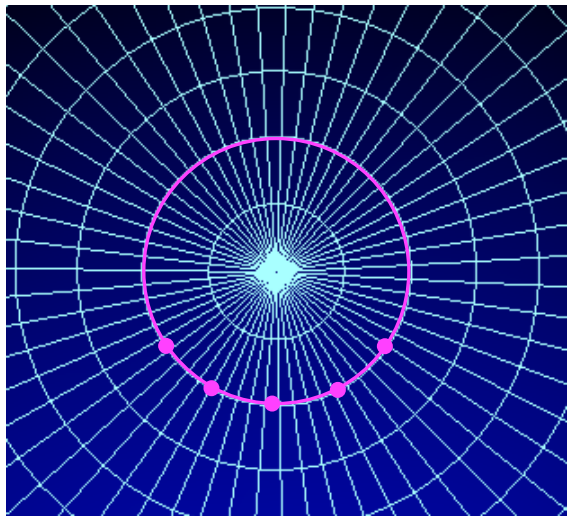
Severe CFL condition (short time step) near the poles.

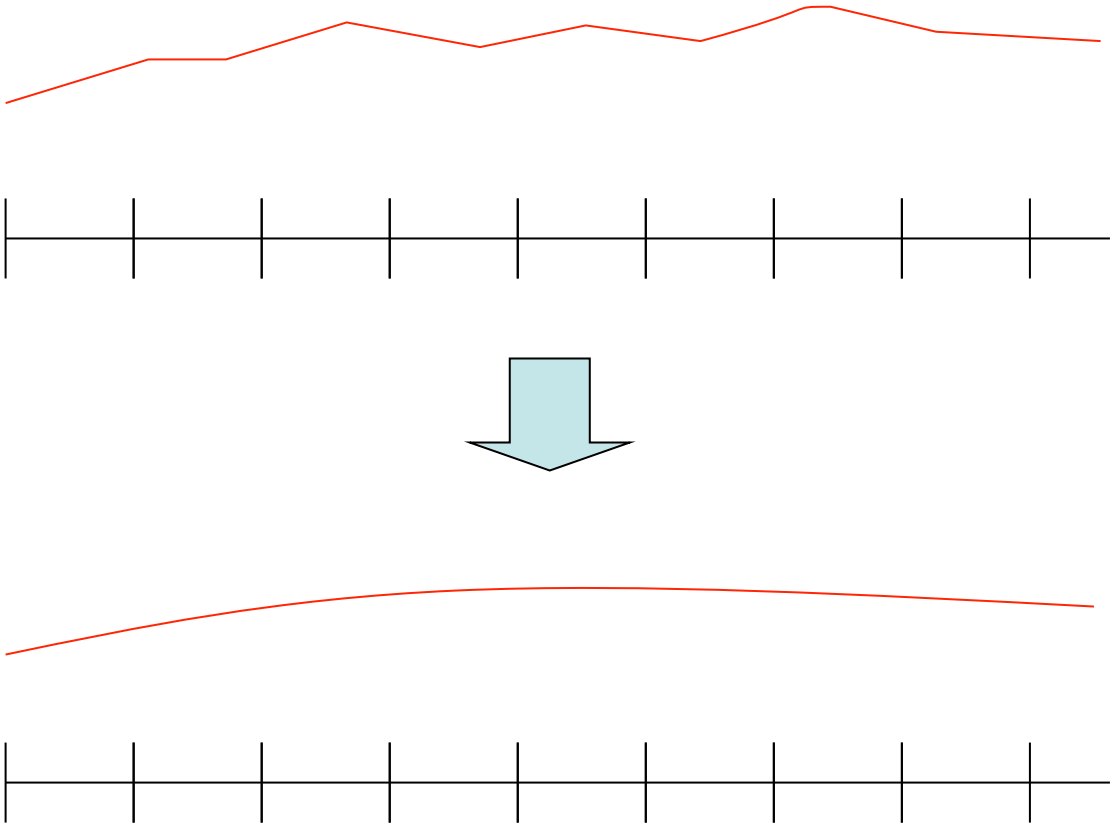


Spherical Filter

Retain the grids, but drop useless high wave number modes.

→ Filtering





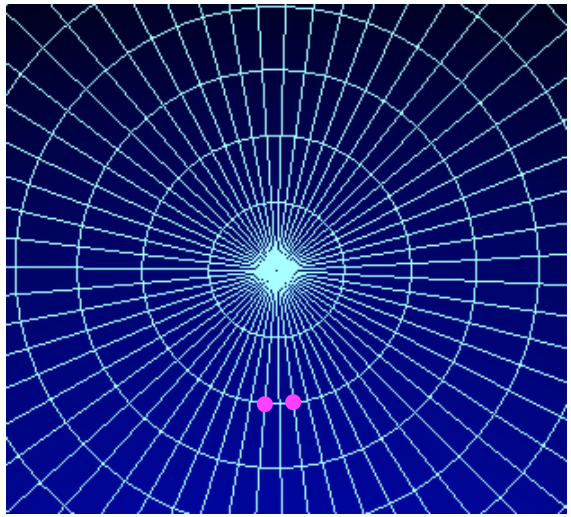
Inefficiency of Lat-Lon Grid

- Too many useless grids in high-latitudes.

  (1) Place many grid points near the poles.
  (Spoiling the low-latitude's resolution.)

  (2) Work hard to calculate data on the grids.

  (3) Throw away most of the data!

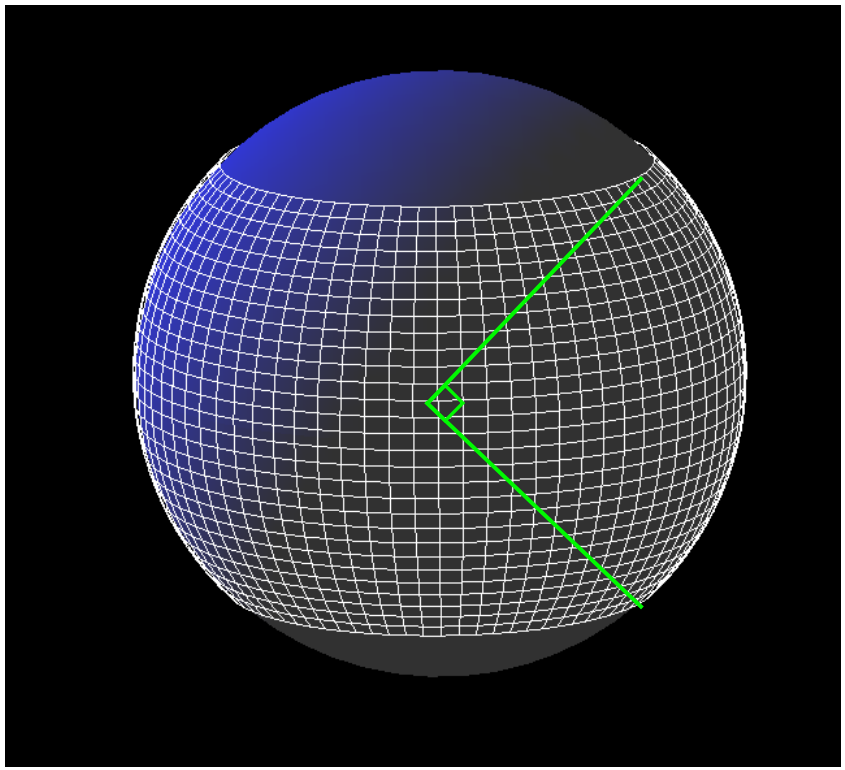


This is true for other spherical discretization methods:
- Double FFT spectral method (FFT both in latitude & longitude).
- Single FFT, hybrid method (FD in latitude & FFT in longitude).

Re-view the latitude-longitude grid

It is almost an ideal grid in the *low latitude region*.

- It is orthogonal coordinates (simple metrics)
- Nearly uniform grid spacing



Overset grid method applied to a sphere

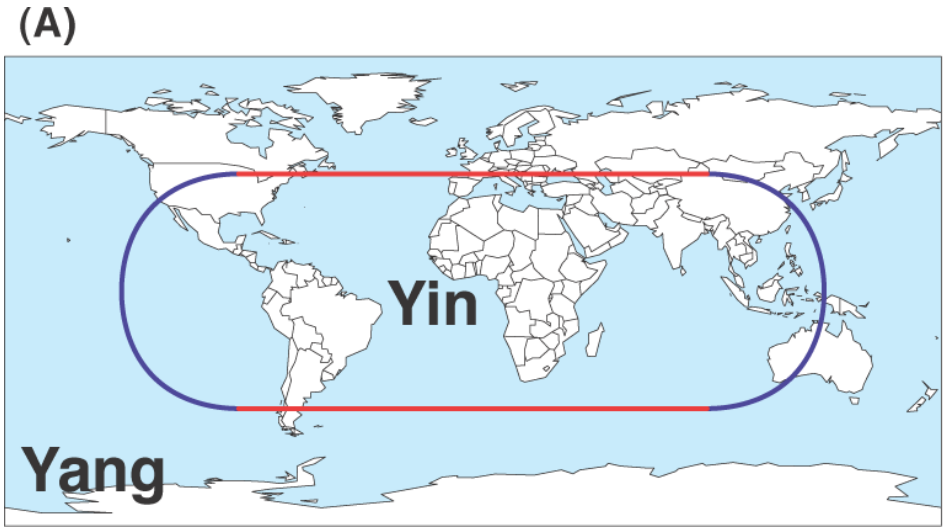
- What is the simplest overset grid on a sphere?
  - Number of component grid = 2
  - The two component grids are the same

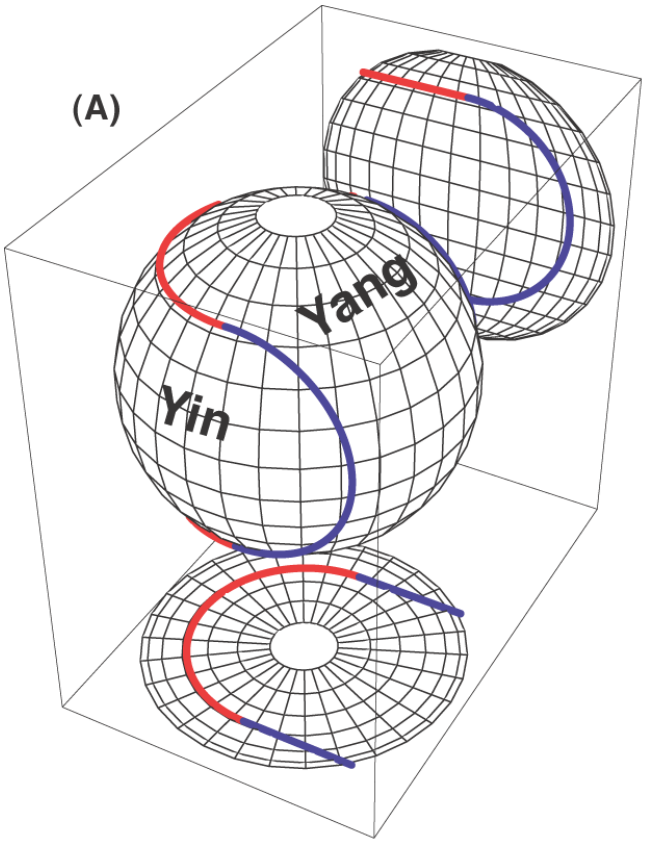
A baseball (or tennis ball)

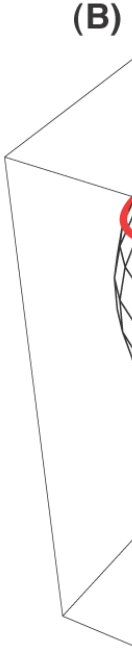
A spherical surface is covered by

- combination of two identical parts (patches).
- one seam.

Dissections of a sphere into two identical parts

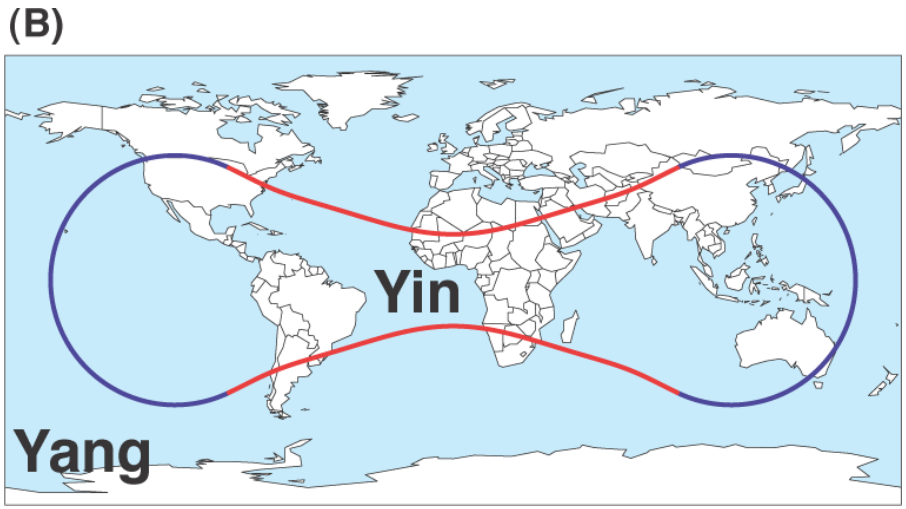


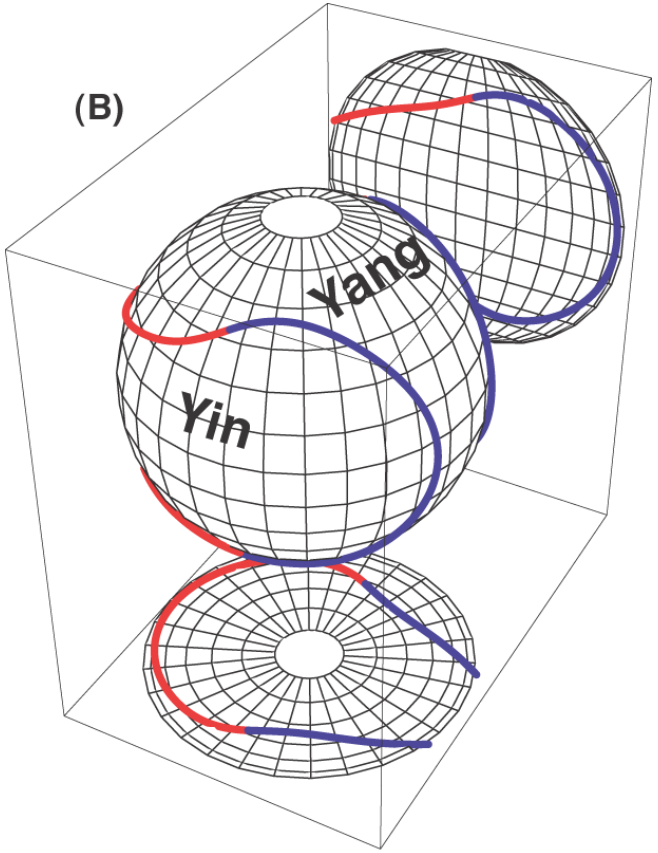




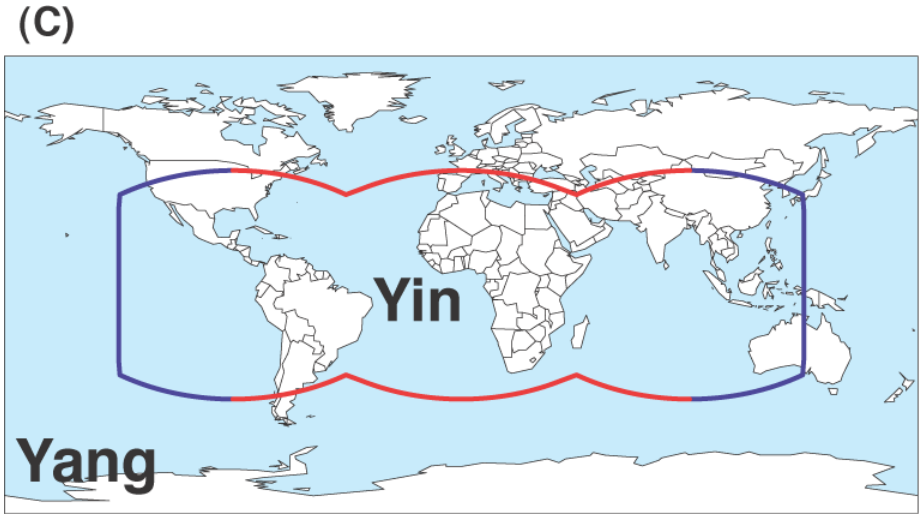
Yin-Yang 陰陽

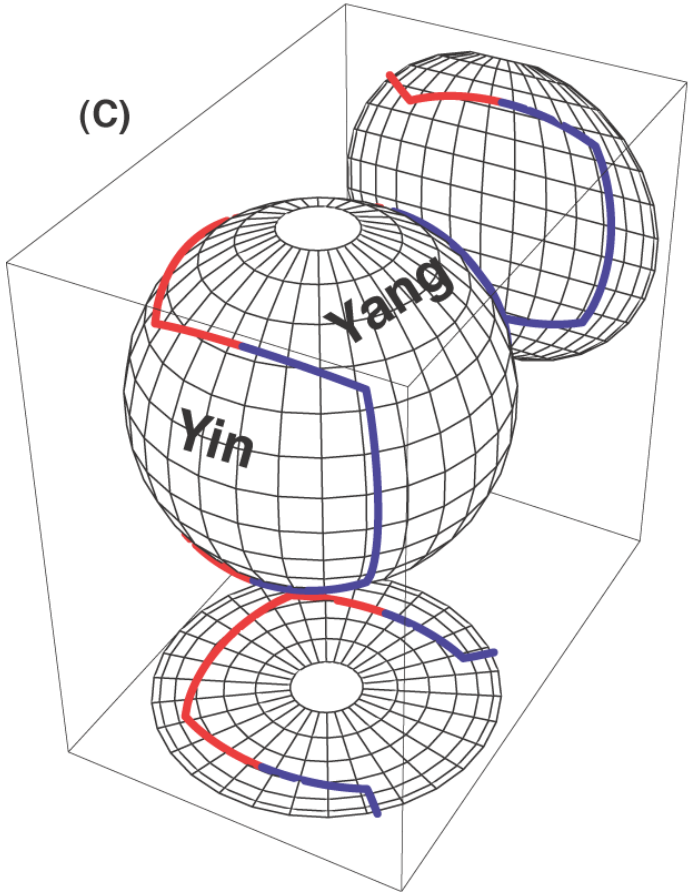
Dissections of a sphere into two identical parts



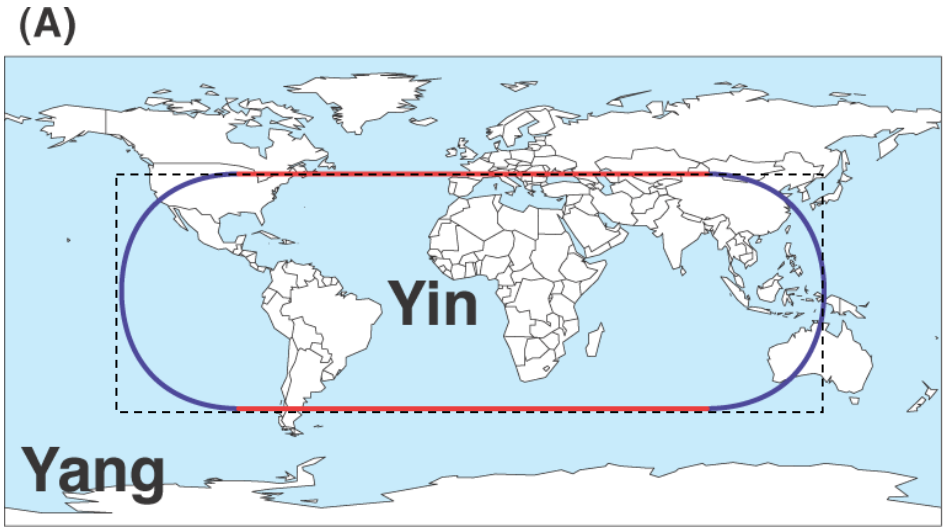


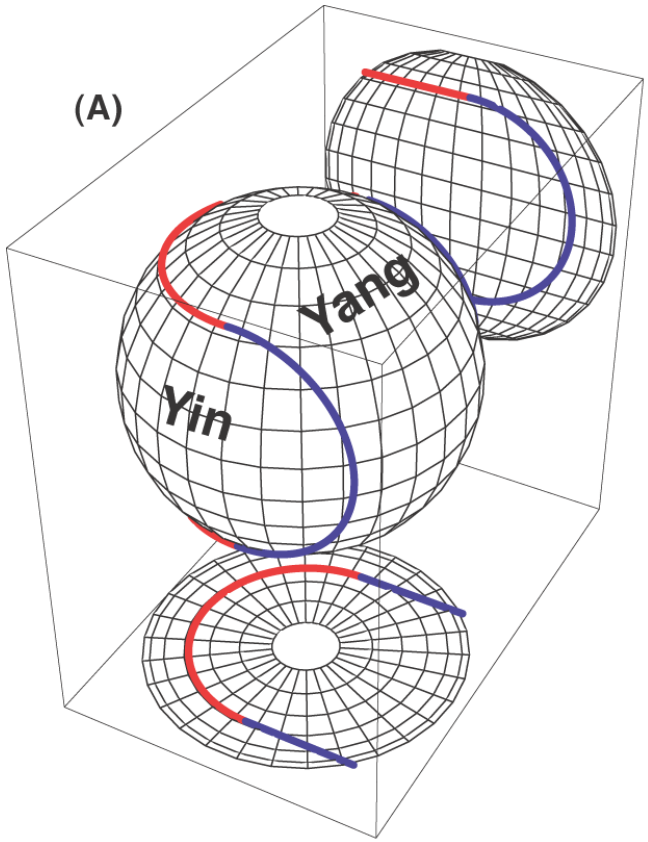
Dissections of a sphere into two identical parts

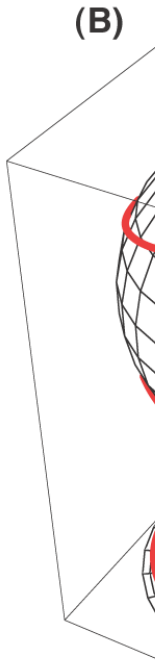




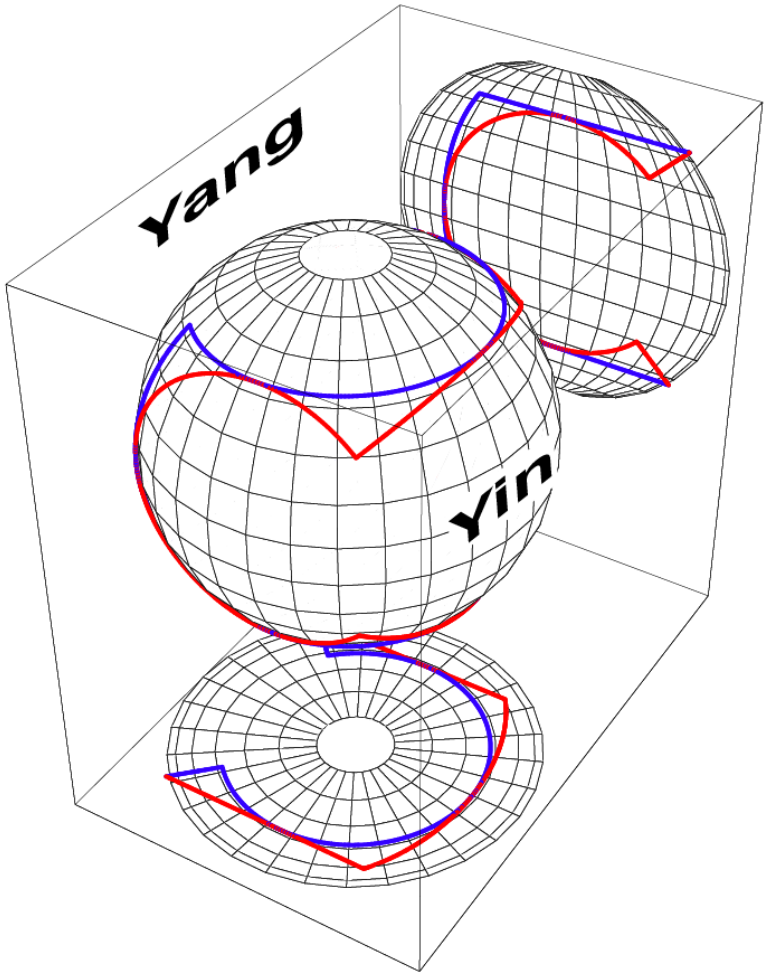
Dissections of a sphere into two identical parts



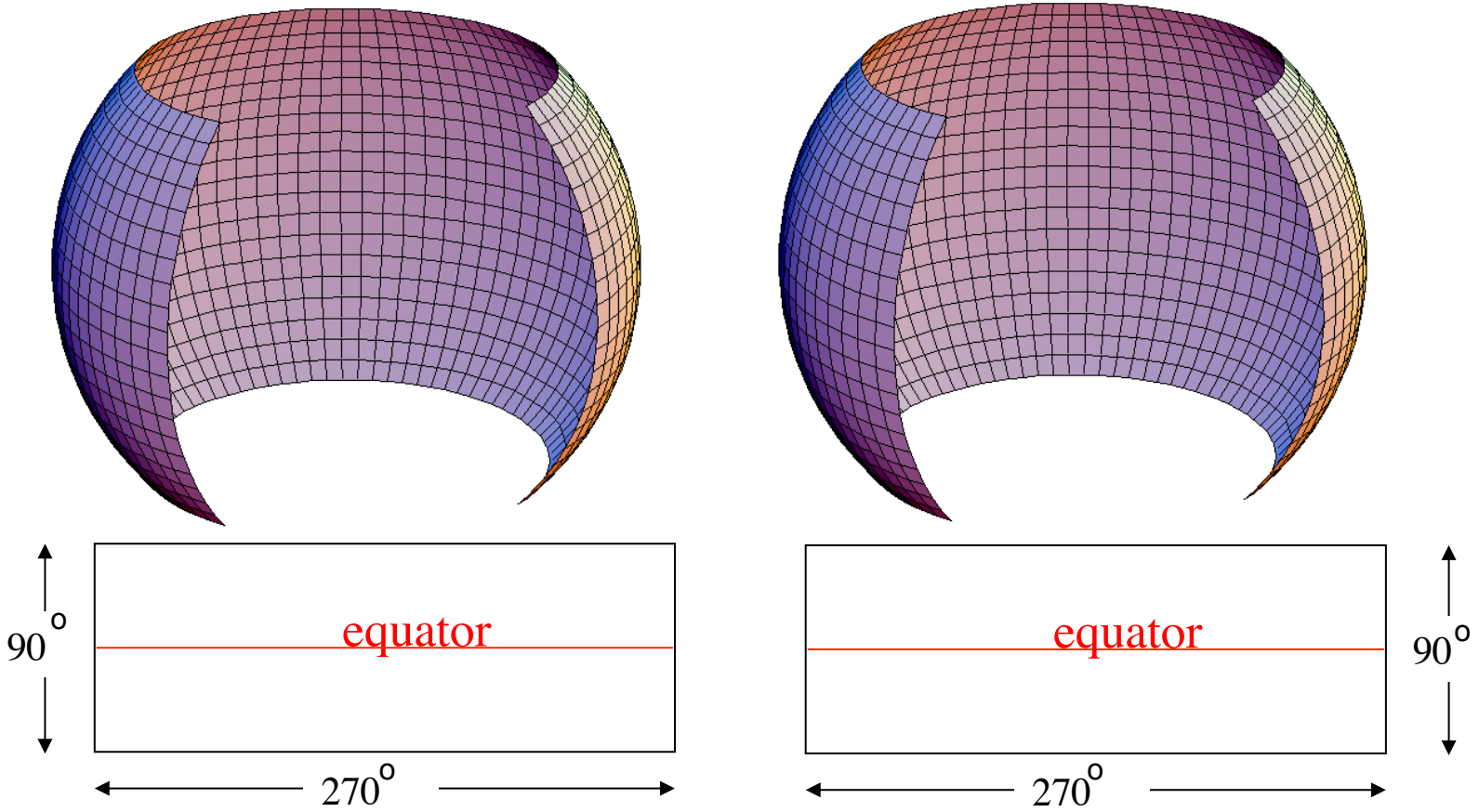




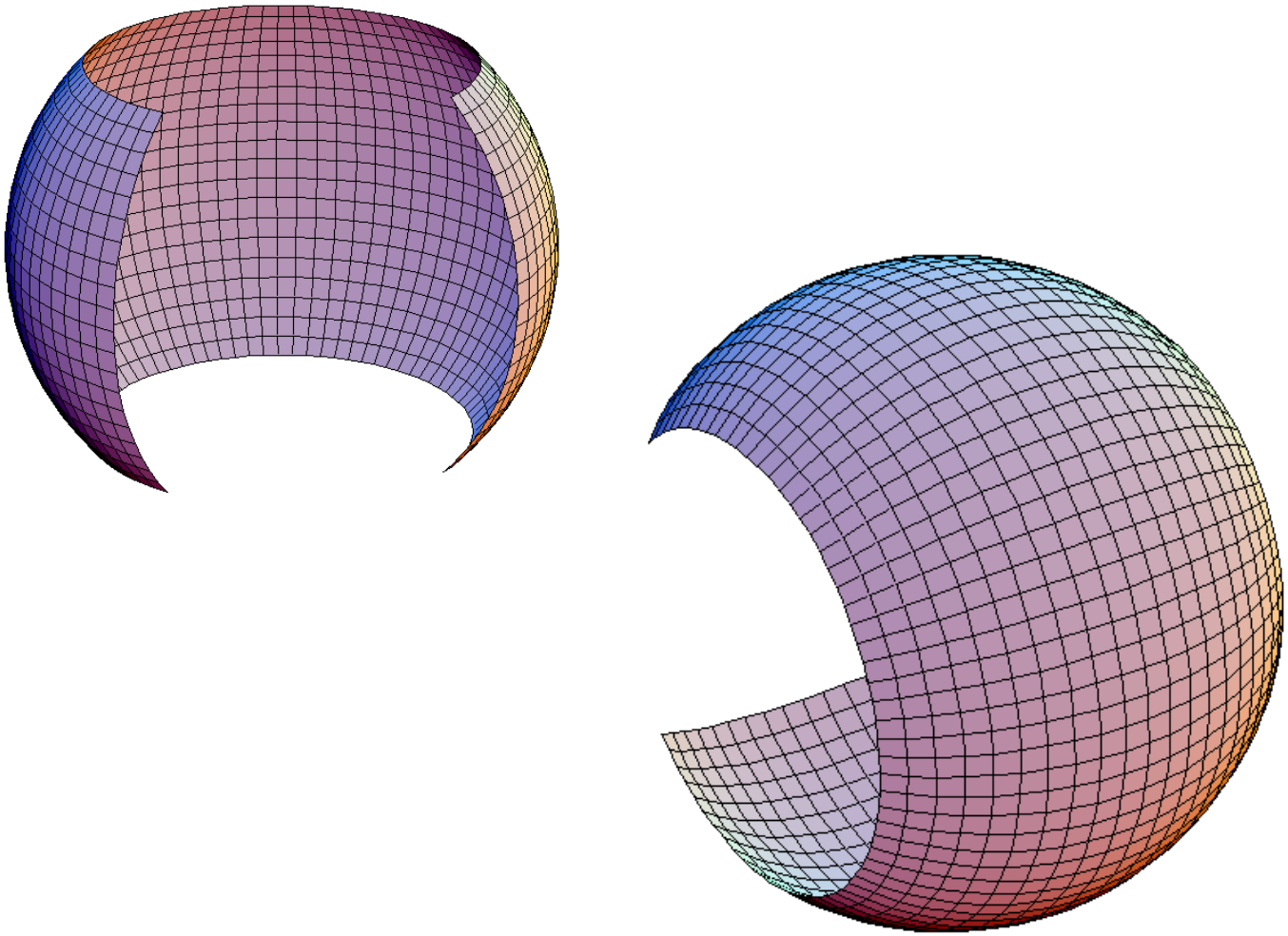
Spherical dissection with partial overlap



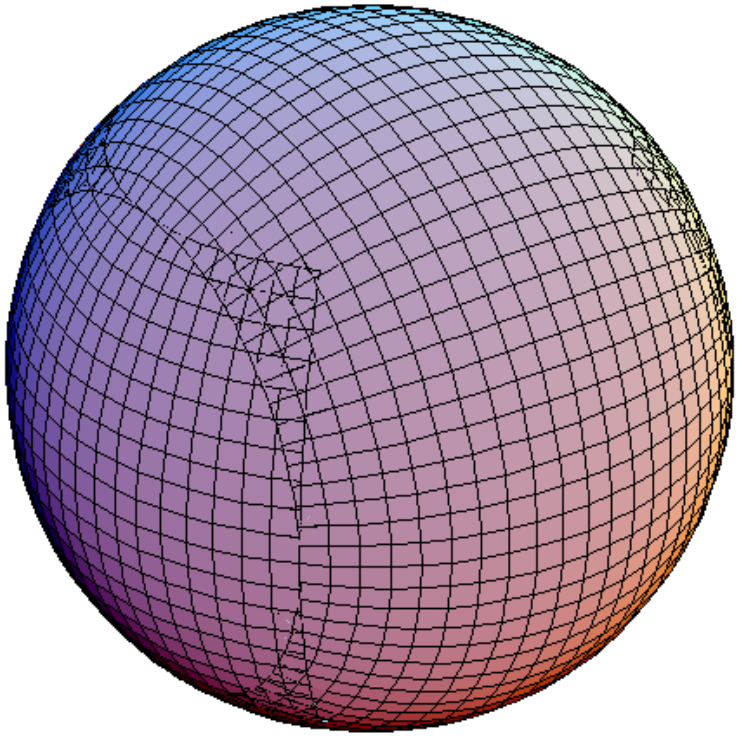
Combining two identical sub-grids to cover a full sphere



Combining two identical sub-grids to cover a full sphere



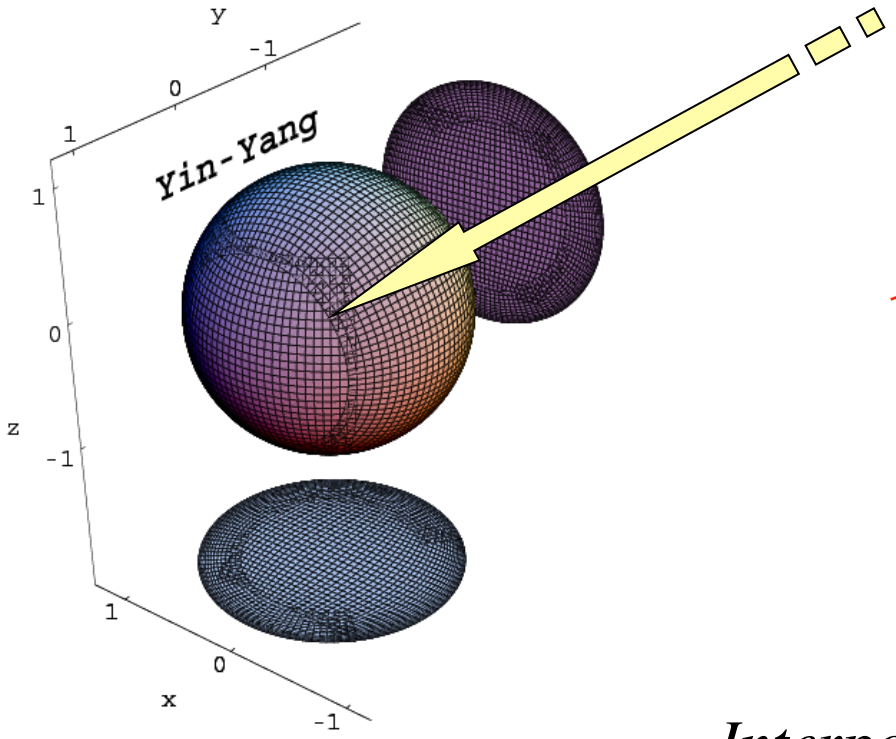
Yin-Yang grid

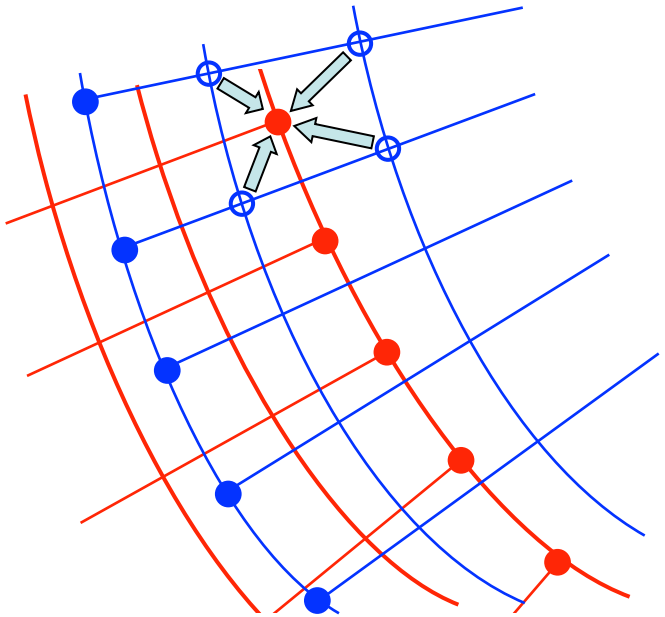


Yin-Yang **陰陽**

Yin-Yang grid as an overset grid

Partial overlap between Yin and Yang grids.



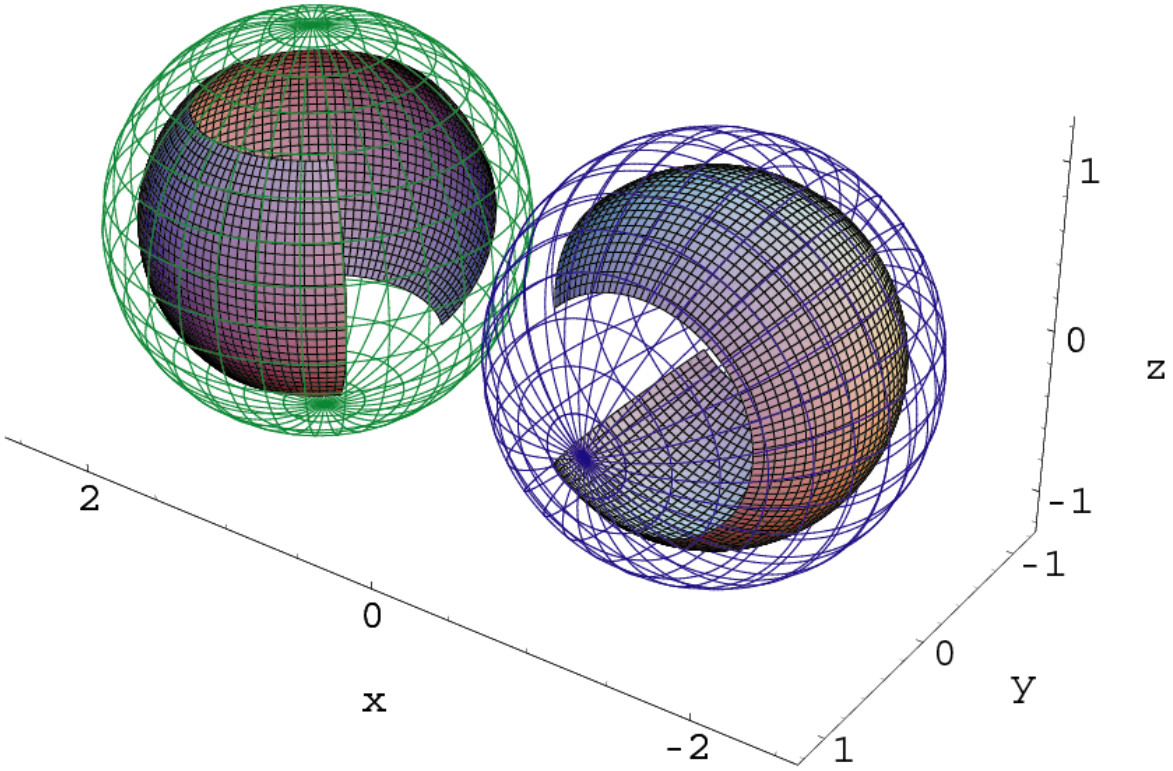


Interpolation

boundary condition

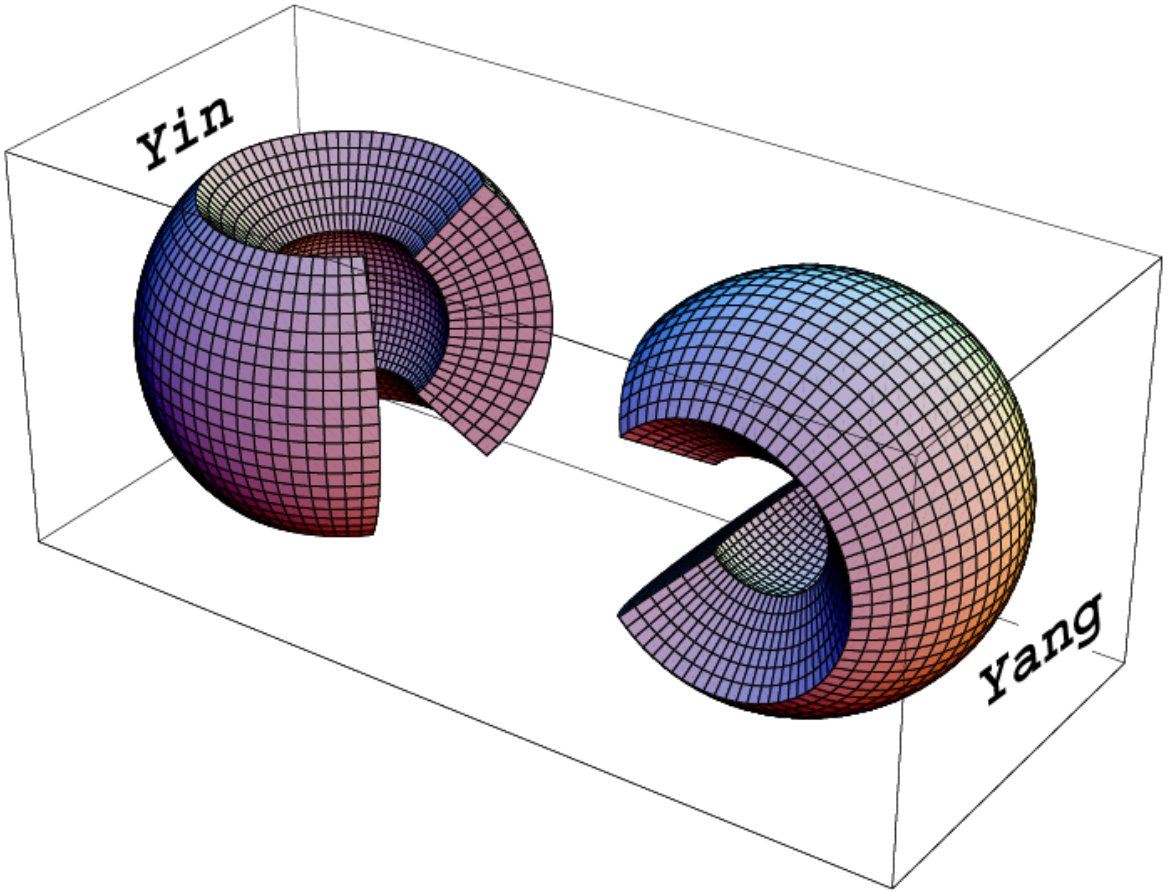
Concise coding of Yin-Yang grid:

- Make one routine on the (partial) latitude-longitude grid.
- Recycle it for two times; one for Yin and another for Yang.



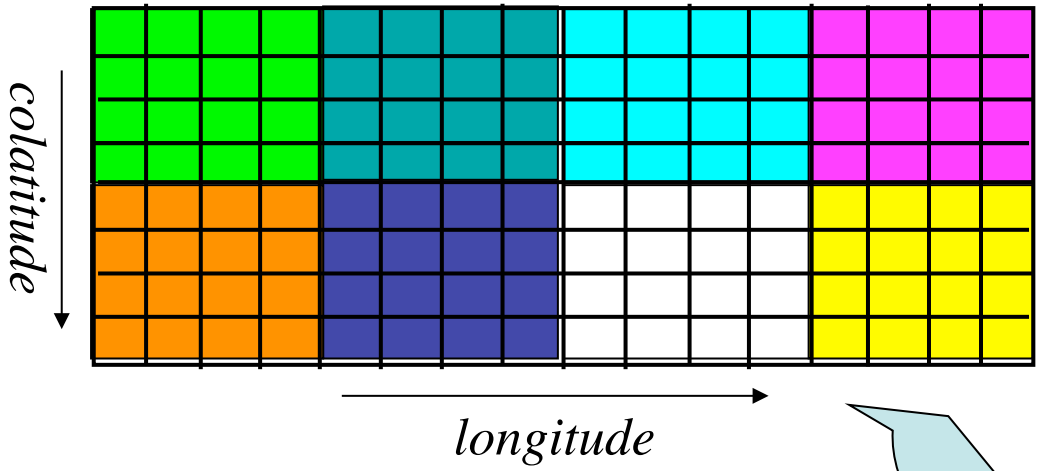
Routines for
- MHD solver
- boundary conditions
- interpolations

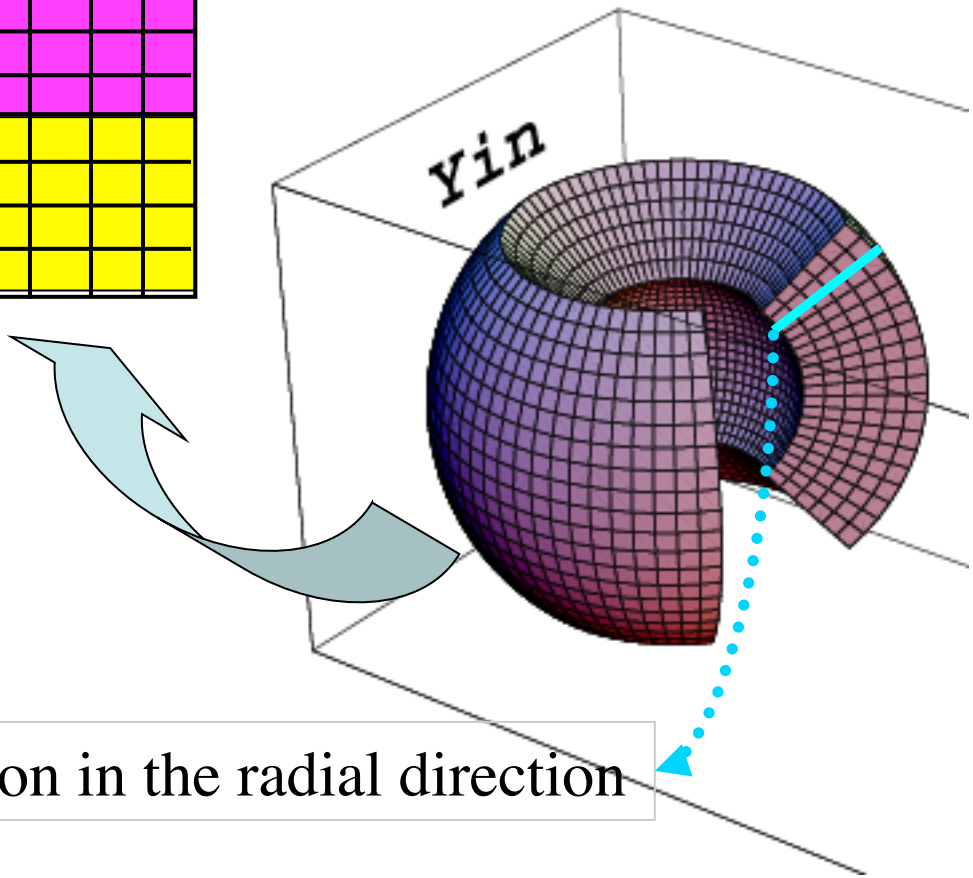
3-D Yin-Yang grid for spherical shells



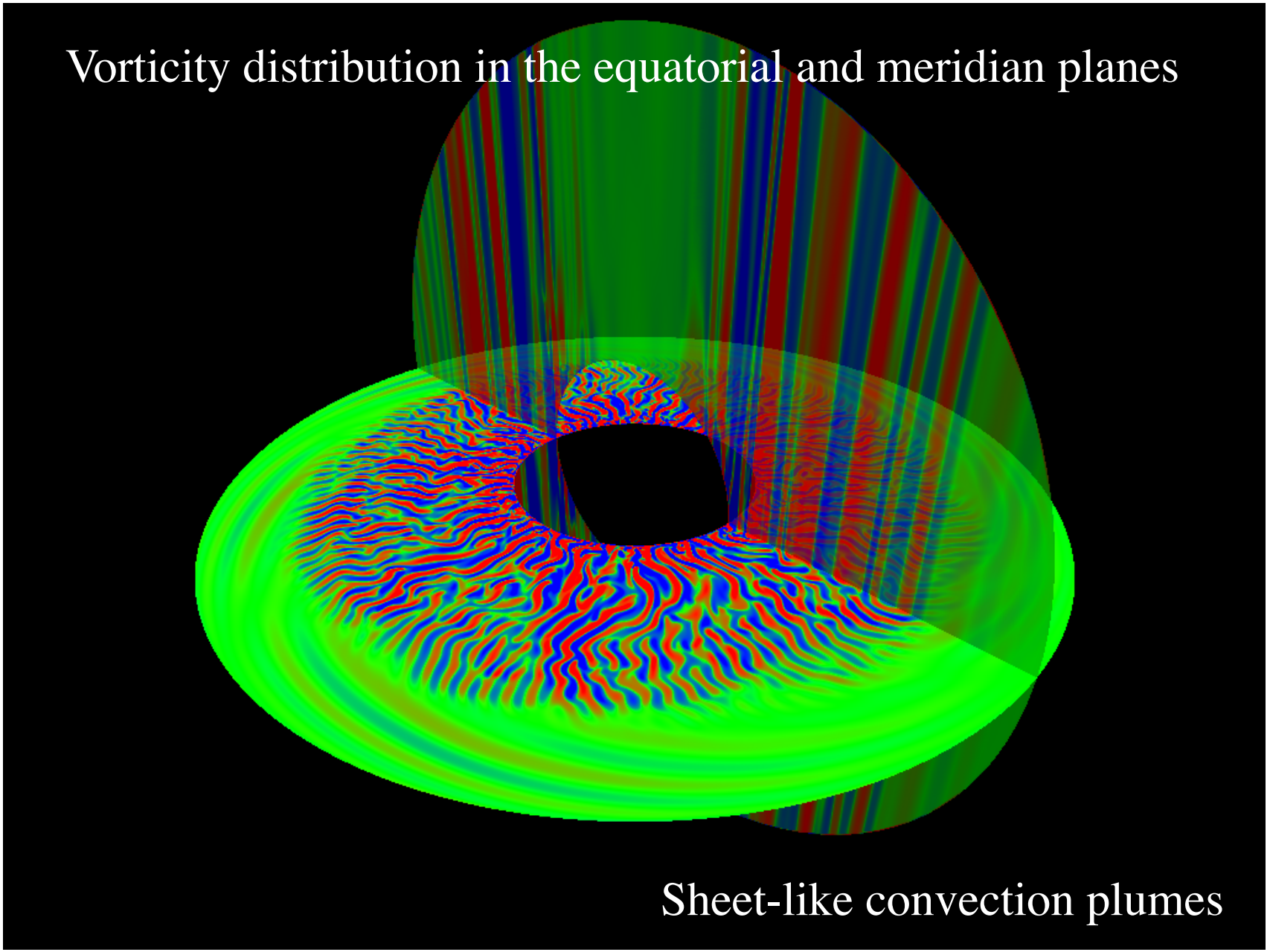
Parallelization on the Yin-Yang grid

2-dimensional domain decomposition in the horizontal computational space.





Vectorization in the radial direction


Vorticity distribution in the equatorial and meridian planes
Sheet-like convection plumes

Summary

- Basic ideas of MHD simulation
- Many simple/sample/example codes.
- I'll make the source codes available for you.
- Check my lab's website:
  URL: http://www.research.kobe-u.ac.jp/csi-viz/